ALGORITHM-BASED EFFICIENT APPROACHES FOR MOTION ESTIMATION SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Teahyung Lee

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
December 2007

ALGORITHM-BASED EFFICIENT APPROACHES FOR MOTION ESTIMATION SYSTEMS

Approved by:

Dr. Russell M. Mersereau
Professor, School of ECE
Georgia Institute of Technology

Dr. Donald Scott Wills
Professor, School of ECE
Georgia Institute of Technology

Dr. David V. Anderson, Advisor
Professor, School of ECE
Georgia Institute of Technology

Dr. Alexander Gray
Professor, College of Computing
Georgia Institute of Technology

Dr. Xiaoli Ma
Professor, School of ECE
Georgia Institute of Technology

Date Approved: Nov 2007

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This research addresses algorithms for efficient motion estimation systems. With the growth of wireless video system market, such as mobile imaging, digital still and video cameras, and video sensor network, low-power consumption is increasingly desirable for embedded video systems. Motion estimation typically needs considerable computations and is the basic block for many video applications. To implement low-power video systems using embedded devices and sensors, a CMOS imager has been developed that allows low-power computations on the focal plane. In this dissertation efficient motion estimation algorithms are presented to complement this platform.

In the first part of dissertation we propose two algorithms regarding gradient-based optical flow estimation (OFE) to reduce computational complexity with high performance. The first is a checkerboard-type filtering (CBTF) algorithm for prefiltering and spatiotemporal derivative calculations. Another one is spatially recursive OFE frameworks using recursive LS (RLS) and/or matrix refinement to reduce the computational complexity for solving linear system of derivative values of image intensity in least-squares (LS)-OFE. From simulation results, CBTF and spatially recursive OFE show improved computational efficiency compared to conventional approaches with higher or similar performance.

In the second part of dissertation we propose a new algorithm for video coding application to improve motion estimation and compensation performance in the wavelet domain. This new algorithm is for wavelet-based multi-resolution motion estimation (MRME) using temporal aliasing detection (TAD) to enhance rate-distortion (RD) performance under temporal aliasing noise. This technique gives competitive or better performance in terms of RD compared to conventional MRME and MRME with motion vector prediction through median filtering.

By considering our proposed algorithms intelligently, algorithm-based efficient motion estimation systems for low-complexity with high performance can be implemented.

DEDICATION

To God, my parents, and my lovely wife

for their constant encouragement, support, and selfless love

ACKNOWLEDGMENT

I would like to thank everyone who helped me get through all the difficulties and accomplish this Ph.D.

First, I would like to express my deepest thanks to my advisor, Dr. David V. Anderson, for his guidance, support, and friendship during my Ph.D. research journey. He has always been supportive not only when I made progress but also when I made mistakes. I learned everything that need for a professional career from him: how to decide under difficult situations, how to write papers, even how to deal with people and make a good relationship. It was my privilege to finish my Ph.D. under his guidance. I would also like to express my heartfelt gratitude to my dissertation committee members, Dr. Russell Mersereau, Dr. Xiaoli Ma, Dr. Scott Wills, and Dr. Alexander Gray for their constructive comments, suggestions, and interactions when I need those. I would like to express my gratitude to Dr. Paul Hasler for his vision and research direction in the Co-operative Analog and Digital Signal Processing (CADSP). His pioneer spirits have led me to explore my Ph.D. research in the CADSP context.

I was fortunate to be a member of the CADSP/EPS research group. Great research and social atmosphere among colleagues makes me focus on my research without losing my interest. I specially thank Sourabh Ravindran, Sunil Kamath, and Shyam Subramanian for their friendship and advice when it comes to writing papers and/or job search. And I am thankful to Ken Chiu, Ryan Robucci, Jungwon Lee, and Brian Gestner for their constructive comments and contribution on my research progress. I would also like to thank the other members of CADSP/EPS and CSIP research groups for their support and companionship. They made my life at Georgia Tech pleasing and enjoyable. I am reluctant to call them as group members since they are more like friends.

My Korean friends in Georgia Tech and Korea University Alumni in Georgia Tech were all my supporting grounds. I specially thank Taeweon Suh, Sungrae Cho, Hasung

Kim, Tae-Young Chang, and Chang-Ho Lee.

I can not forget our small group members and anonymous brothers and sisters in Saehan Presbyterian Church. Their constant fellowship and prayers gave me a great strength in good times and bad times.

I would like to thank my parents, parents-in-law, brother, and his family for everything they have done for me. Without their encouragement, unconditional love, support, and prayers, I would not have been able to meet the challenges of this research.

I do not know how to describe my deepest grateful feeling in words for my lovely wife, Wooyoung Sung. She is always a great source of strength whenever I have met some troubles. In addition, she makes my joy double and agony in half. Because of her endless support and endurance, I finally finished my Ph.D. life. This work can not be available in the world without my lovely wife.

Last but not least, I thank my Lord, Jesus Christ, who is leading my life and gives me the meaning and purpose to live in this world.

CHAPTER 1

INTRODUCTION

Motion information is crucial for many video processing tasks such as video coding, shape reconstruction, autonomous navigation, denoising, and object tracking. However, extracting or estimating motion information can be computationally burdensome. As video frame rate and/or video format size per each frame increase, finding efficient methods to extract motion is important for effective system design. In addition, wireless device applications need power-efficient systems with high performance. However, power consumption is largely a function of the implementation technology and therefore, difficult to analyze in general. Instead, we consider computational complexity and, where appropriate, memory access because these can be used to calculate power on a particular platform. In this research we explore efficient algorithm approaches for motion field estimation systems with respect to computational complexity, memory access, and performance.

In this dissertation, we propose the rapid algorithm verification methodology for a co-operative analog-digital signal processing (CADSP) approach to predict possible performance degradation. As an efficient filtering algorithm and structure for gradient-based OFE system, checkerboard-type filtering (CBTF) is proposed. We will present the way in which our checkerboard-type filtering can reduce total computational complexity based on the number of operations and memory access in three-dimensional (3-D) pre-filtering and gradient-operator for gradient-based OFE in the following sections. And algorithmic efficiencies for OFE are achieved with new low-complexity OFE algorithms using spatially recursive least-squares (RLS) technique with adaptive filtering and/or matrix refinement. For motion estimation algorithm in video coding, we propose a new MRME algorithm using multi-dimensional filtering property to achieve better RD performance than previous MRME algorithms.

In the computer vision and video processing areas, an optical flow field, which is a

dense motion flow field, is essential for processing a sequence of images. The applications of the estimated optical flow range from video compression to 3-D surface structure estimation and active exploration. Generally, popular optical flow estimation methods can be classified into gradient-based and correlation-based methods. The latter use a block-wise similarity of intensity values between current and previous frames, and the former solve the linear equations of spatio-temporal derivatives [5]. The motion estimation algorithms in video coding can be regarded as a special case of correlation-based OFE algorithm with spatial searching scheme.

To enhance system efficiency, we apply a cooperative analog-digital signal processing (CADSP) approach for the motion estimation system using the gradient-based optical flow estimation (GBOFE) method. The general goal of a CADSP approach is to reduce power by using both analog and digital computations. To make successful CADSP systems, algorithm and system designers should consider the performance degradation due to sensor and analog circuitry noise and distribution in the processing of an algorithm. Because of noise and distortion issues, assigning regular and expensive computations, which are less sensitive to bit resolution, to the analog domain is desirable instead of to the digital domain [23], [35]. Therefore, filtering steps of a GBOFE method are good candidates for analog domain processing.

The development time of an analog system can be much longer than that of a digital system or algorithm implementation. To allow rapid hardware and software co-verification for a transform imager, a CADSP-domain imager for 2-D FIR filtering and block transformations, a software simulation tool is developed. For the imager designer, a simulator is employed to validate outputs of the transform imager under various conditions. For the algorithm developer, the simulator can be utilized to test and evaluate different algorithms under certain non-ideal factors of the imager. We created a MATLAB-based simulator for this purpose. The simulator captures both the behavioral simulation and the physical device characterization. We use the simulator to validate algorithms for a fast design cycle

while taking the characteristics of the transform imager into consideration. Therefore, the algorithm developer can verify their worst- and best-case algorithm performances in this imaging platform.

As an efficient filtering algorithm and architecture, we suggest a checkerboard-type filtering algorithm for gradient-based optical flow estimation. This algorithm achieves maximal parallel structure for prefiltering and derivative filtering, which can lead to reduce the number of memory accesses and computational complexity with high performance. This scheme is well-suited to both SIMD and a focal plane pipelined architectures.

Based on simulation results in [44] and [48], the simplified Lukas-Kanade (LK) algorithm, which is a GBOFE algorithm using the least-squares (LS) technique with weighting, is one of the most successful solutions in terms of noise-robustness and the number of operations. However, it still has some redundancies for calculating successive LSs among adjacent pixels. Therefore, recursive least-squares (RLS) techniques can be applied for improving the computational efficiency. In this research, we show the results of performance and computational complexity among standard LS-OFE and spatially recursive LS-OFE algorithms using adaptive filtering and/or matrix refinement.

To improve the rate-distortion performance in video coding, we suggest a new MRME algorithm using temporal aliasing property. Because of the spatial-variant property and relatively high speed of objects compared to the frame rate and image size, temporal aliasing is unavoidable in conventional MRME. By using the spatial-temporal filtering property, we can detect the temporal aliasing with some confidence. Our new algorithm can utilize the temporal aliasing information efficiently to improve the coding gain compared to the conventional MRME algorithms. MRME is much more efficient than FSBM but it produces suboptimal results due to aliasing. Our approach decreases the computational efficiency of MRME but reduces the aliasing problems and is still more efficient than FSBM.

1.1 Contributions of the Dissertation

In this dissertation, algorithmic efficient motion estimation system design is explored with respect to computational complexity, memory access and/or performance. To achieve this goal, some performance tests under various noisy environments, and a new algorithm verification methodology are introduced for CADSP system design. A CADSP approach gives an opportunity for reducing power-consumption by performing signal processing using analog system in sub-threshold region. However, the digital processing should be also considered to improve the efficiency of overall system, which is motivates this research. Therefore, we employ sophisticated signal processing ideas to enhance system design parameters such as the number of memory accesses, computational complexity, and performance. For example, one of novel achievements in this dissertation is checkerboard-type filtering (CBTF) for reducing computational complexity and the number of memory accesses under CADSP system design. The main contributions of this dissertation include:

1. Simulations on gradient-based LS optical flow estimation algorithm performance under various sensor and system noises to verify the reasonable linear-range and/or the number of bit resolutions for CADSP system design;

2. Rapid algorithm verification methodology for reducing hardware and software algorithm co-verification time of the system under CADSP design concept by using a CADSP imager simulator incorporating physical and architectural modelings;

3. A novel filtering structure for the prefiltering stage of GBOFE algorithms;

4. An efficient spatially recursive least-squares (RLS)-based OFE algorithm framework for reducing computational complexity exponentially as the number of motion modeling parameters increase;

5. Analysis of trade-offs for a CADSP OFE prefiltering structure; and

6. A novel wavelet-based MRME algorithm to improve rate-distortion performance by using temporal aliasing detection based on spatial-temporal filtering properties.

1.2 Dissertation Organization

The rest of this dissertation describes motion estimation system design in terms of computational complexity, the number of memory access, and performance. The following chapter provides previous and background work regarding image sensors and popular optical flow estimation algorithms. In addition, some existing approaches for OFE and motion estimation, and the relationship between two algorithms are described. In Chapter 3, OFE algorithm performance under noisy environments is described. This work indicates the expected performance and phenomenon of optical flow estimation under various sensor and system noises. Also, a rapid algorithm verification methodology for CADSP imaging system is presented by using architecture-level simulator in chapter 3. This methodology helps to estimate the performance before full implementation of physical system. The checkerboard-type filtering (CBTF) algorithm is introduced in Chapter 4. The advantage of CBTF is to reduce computational complexity and the number of memory access with high performance than those of conventional filtering strategy. The spatially recursive LS-based OFE algorithm frameworks using adaptive filtering and/or matrix refinement are presented in Chapter 5. This work reduces the computational complexity compared to the conventional LS-OFE algorithm. In Chapter 6, a new MRME algorithm using temporal aliasing detection is suggested to improve the rate-distortion performance compared to that of conventional MRME. Final conclusions about the dissertation and future research directions are described in Chapter 7.

CHAPTER 2

BACKGROUND AND PREVIOUS WORKS

In this chapter, we review previous research on algorithms and systems related with optical flow estimation (OFE) and motion estimation. This chapter is divided into four sections. Imaging system design is presented in section 2.1. Optical flow estimation algorithms are described in section 2.2. Motion estimation for video processing is presented in section 2.3. Multi-dimensional filtering for motion estimation is explained in section 2.4. Finally, low-power OFE system design is discussed in section 2.5.

2.1 Imaging System Design

When considering image/video processing systems such as OFE and filtering operations, it is always advantageous to start with the understanding of the imager. In this subsection, we cover the general architecture of CCD and CMOS image sensors.

2.1.1 CCD and CMOS Image Sensors

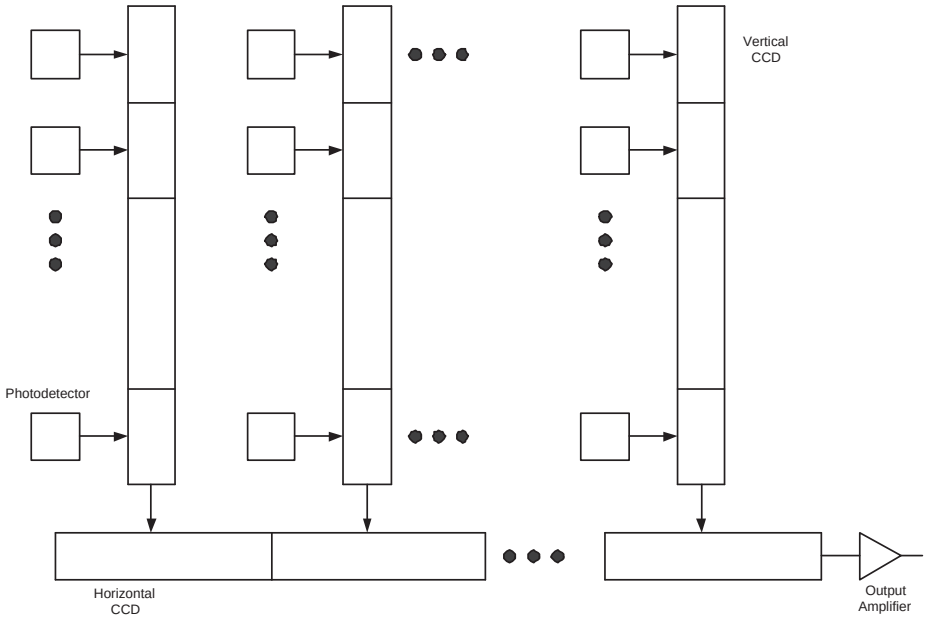


Figure 2.1. The block diagram of a typical CCD image sensors [42].

As more and more people have gained access to the rapidly expanding internet and mobile services and the ever increasing personal computing power, digital cameras and video camcorders are becoming very popular. The most widely used imaging sensor technique is the charge-coupled devices (CCDs) [30], [26]. Figure 2.1 depicts the block diagram of the widely used interline transfer CCD image sensors. In the CCD image sensors, incident photons are converted to electric charges which are then accumulated by the photo detectors during exposure time. The charges are simultaneously transferred to vertical CCDs for all the pixels at the end of the exposure time. During the following readout time the accumulated charge is sequentially transferred into the vertical and horizontal CCDs, and finally shifted to the chip level output amplifier, where it is converted to voltage signal. As the dominant image sensor technology used by digital cameras, CCD image sensors achieve superior noise performance and uniformity. The reason that CCD image sensors have low noise and high uniformity is that they are fabricated using specialized processes with optimized photo detectors. However, they cannot be integrated with other analog and digital circuits such as memory which are typically implemented in CMOS technology. Even worse, because CCDs are high capacitance devices and all the CCDs are switched at the same time with high voltages during readout, CCD image sensors usually consume considerable power. Therefore, normally CCD image sensors based digital cameras and video camcorders are relatively large in size and consume high power, and thus are not well suited for portable and embedded imaging applications.

In contrast, CMOS image sensors are fabricated using standard CMOS process with no or only minor modifications [20], [77]. This leads to one of the most important trends in digital camera design, which is the use of CMOS image sensors instead of CCDs as the imaging devices. Figure 2.2 plots the block diagram of a typical CMOS image sensor. Unlike CCD image sensors, it adopts digital memory style readout, using row decoders and column amplifiers. Current CMOS image sensors typically have lower image quality and higher noise level than CCD image sensors, mainly because the fabrication process

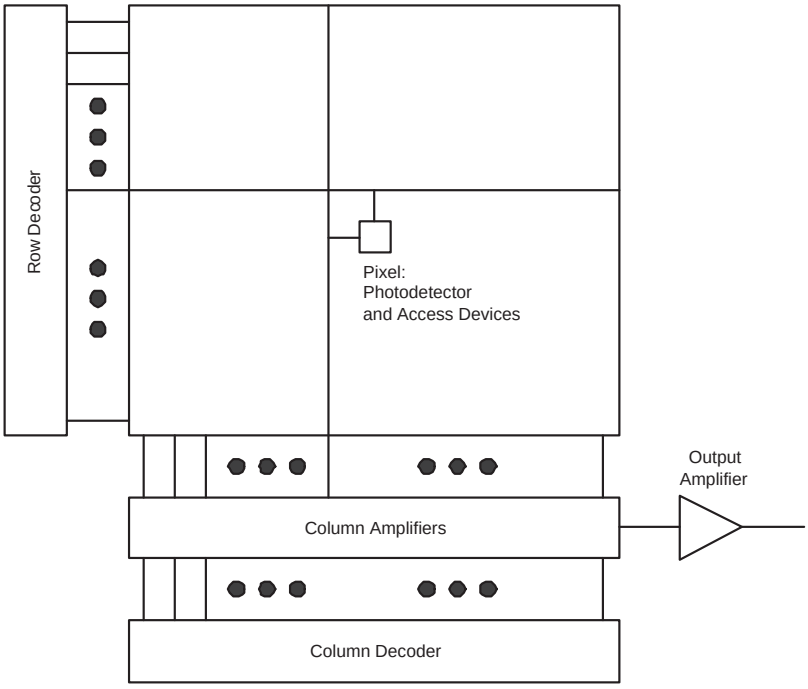


Figure 2.2. The block diagram of a typical CMOS image sensors [68].

is not optimized for image sensing. Moreover, CMOS imagers have higher fixed pattern noise (FPN) since image data are read out through different chains of buffers and amplifiers (Fig. 2.2). However, a digital memory style structure helps CMOS to overcome many of the problems that CCD image sensors have. First, CMOS image sensors consume much less power than CCD image sensors due to lower voltage swing, switching frequency and capacitance. Second, random access of pixel values becomes possible, allowing selective readout of windows of interest. Third, analog signal processing, A/D conversion, and memory can be integrated on a single chip, as demonstrated in [1] and [19].

2.2 Optical Flow Estimation

In this section, we first introduce some basic definitions regarding optical flow. Typically popular optical flow estimation techniques can be grouped into two classes: Differential/gradient-based techniques, which compute the velocity from spatiotemporal

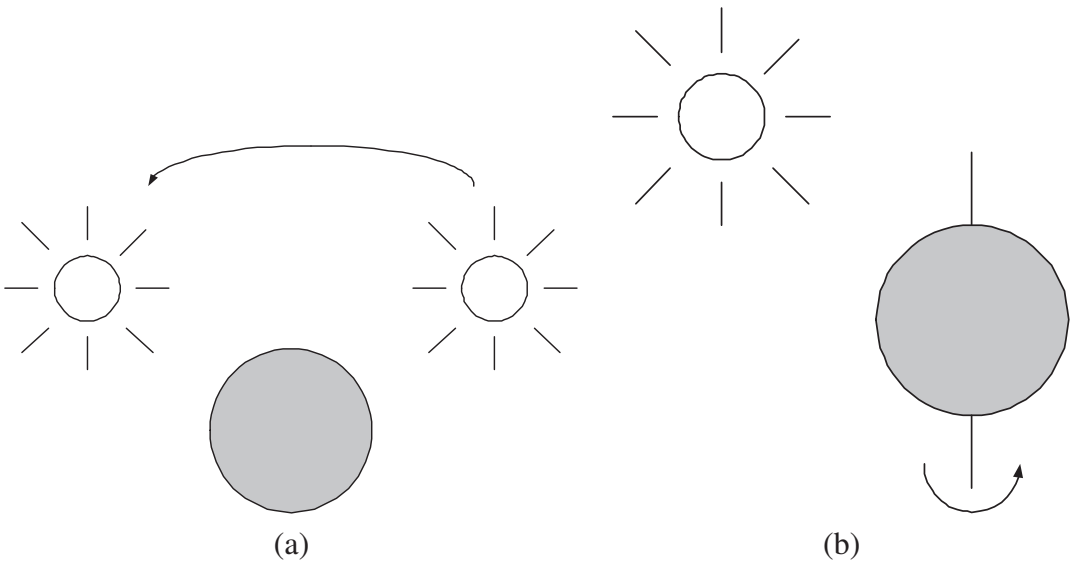


Figure 2.3. Examples of discrepancy between motion field and optical flow field [56]. (a) Optical flow due to motion of the light source; (b) No optical flow due to no change of image intensity

derivatives of the image intensity, and correlation-based techniques, which define the velocity as the shift that returns the best fit between image areas at different instances of time. We discuss motion analysis and describe both optical flow estimation techniques more specifically in this section.

2.2.1 Motion Analysis

In computer vision, motion information is required for a variety of applications, such as visual tracking, structure recovery, robot/vehicle navigation, and more. To understand how accurately estimate motion information from an image sequence, we need to describe some terminology definitions related with motion information/algorithm in computer vision.

Visual motion results from the displacement of the scene with respect to a fixed camera or vice versa. Under those situations, motion field is a 2-D object velocity field that results from a projection of the 3-D scene velocities. In reality, the accurate estimations of a motion field and 3-D scene velocities are very hard to obtain without prior knowledge of the scene [65], [71]. Instead, we can estimate an optical flow, which is a 2-D velocity field describing the apparent motion in an image sequence [65], [29]. Typically optical

flow estimation algorithms exploit the movement of the brightness patterns over the image sequence. This implies that the optical flow can generate false motion field information if brightness consistency is not guaranteed. For example, if a light source for a fixed sphere with uniform color is moving, optical flow will be generated due to the motion of the light source. On the other hand, for the untextured sphere of same color rotating under fixed illumination, there will be no apparent motion and hence no optical flow will be calculated. These examples are presented in Fig. 2.3.

2.2.2 Gradient-based OFE and Simplified LK-OFE

Gradient-based techniques estimate the motion field by using the spatiotemporal derivatives of the image intensity or filtered versions of image (using band-pass or low-pass filters). These techniques have assumption that a point in an image has the same intensity in other successive frames. In other words, the brightness consistency is assumed over time. This assumption for all differential optical flow algorithms motivates a constraint equation, which is derived from the Taylor series expansion of the translational brightness consistency between successive image sequences.

The brightness consistency can be described in Eq. (2.1) when a point with the same intensity moves by δ_x, δ_y in time δ_t.

$$I(x, y, t) = I(x + \delta_x, y + \delta_y, t + \delta_t), \tag{2.1}$$

where $I(x, y, t)$ is the intensity of the image at a point (x, y) at a time t.

By applying a first-order Taylor series expansion on $I(x, y, t)$, we can derive

$$I(x + \delta_x, y + \delta_y, t + \delta_t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t + H.O.T., \tag{2.2}$$

with the H.O.T. representing higher order terms that can be be ignored if those are small enough. By using Eq. (2.1) and ignoring H.O.T., we can rewrite Eq. (2.2) as

$$\begin{aligned} \frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t &= 0, \\ \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} &= 0, \end{aligned}$$

where $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the spatial and temporal derivatives of the image, and $\vec{v} = (v_x, v_y)^T = (\frac{\delta_x}{\delta_t}, \frac{\delta_y}{\delta_t})^T$ is the velocity. These quantities are referred to as the optical flow and/or motion vector.

Based on Taylor series expansion results, the constraint equation using first derivatives with respect to spatial and temporal domains can be summarized in Eq. (2.3) for gradient-based optical flow estimation algorithms [5].

$$\vec{I_s} \cdot \vec{v} + I_t = 0, \tag{2.3}$$

where $\vec{I_s} = (I_x, I_y)^T$ and I_x, I_y, and I_t are spatial and temporal derivatives respectively of an image I, and $\vec{v} = (v_x, v_y)^T$ is a motion vector for a dense motion field, optical flow. Equation (2.3) is derived from the Taylor series expansion of the translational brightness consistency between successive image sequences as follows:

Unfortunately, we cannot find a unique solution for Eq. (2.3) since Eq. (2.3) has two unknowns in one linear constraint equation, which means it is an under-determined system. This problem is called as the aperture problem: The motion of an object is locally ambiguous because we can observe the object motion typically through a local window, aperture. Therefore, different physical motions are indistinguishable if the motions can be regarded as same within the aperture (See Fig. 2.4). In order to solve Eq. (2.3,) further constraints have to be incorporated. Even though there are a variety of variations such as [32], [7], and [75], we will introduce some exemplary algorithms using local and global optimizations. For example, we present the optical flow algorithms of simplified Lucas and Kanade and Horn and Schunk that use different approaches to recover the full motion vector $\overrightarrow{v}$.

To find the unique solution for motion vector, we have to add more constraint terms. Either local or global optimization techniques may be used to calculate motion vectors using

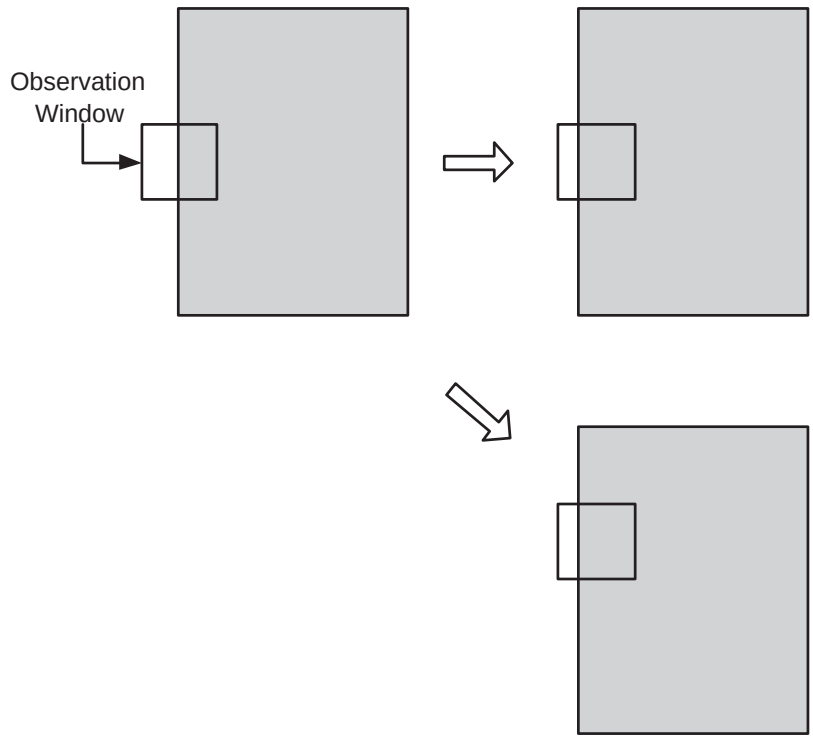


Figure 2.4. The aperture problem. There are no difference in terms of estimated optical flows even though the ground true motion vectors are different.

local or global smoothness constraints, respectively. This can transform the problem into an estimation problem, such as regularization and/or least-squares. To investigate which step of OFE corresponds to the estimation problem, it is necessary to analyze a general gradient-based OFE algorithm in modules. Typically, general gradient-based OFE algorithms can be divided into three steps: (i) perform pre-filtering or smoothing with low-pass/band-pass filters that enhance the signal-to-noise ratio (SNR), (ii) extract spatiotemporal derivatives, and (iii) produce a two-dimensional (2-D) optical flow field by solving the linear equations of derivatives (see Fig. 2.5). The third step of Fig. 2.5 is needed for solving the estimation problem.

Horn and Schunk

Horn and Schunk algorithm combines Eq. (2.3) with a global smoothness constraint term to calculate the velocity field $\overrightarrow{v} = (v_x, v_y)$ [29]. This approach is regarded as regularization. The optical flow is then computed by minimizing

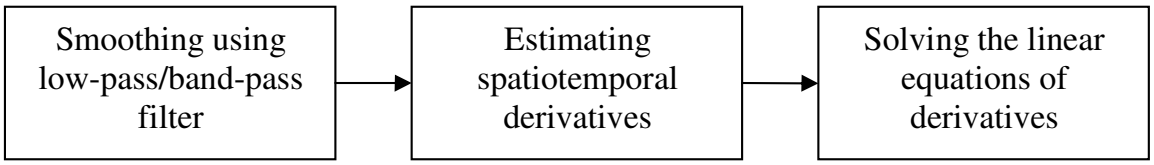


Figure 2.5. The block diagram of gradient-based OFE algorithms.

$$\int_D (\vec{I_s} \cdot \vec{v} + I_t)^2 + \lambda^2 \left[\left(\frac{\partial v_x}{\partial x} \right)^2 + \left(\frac{\partial v_x}{\partial y} \right)^2 + \left(\frac{\partial v_y}{\partial x} \right)^2 + \left(\frac{\partial v_y}{\partial y} \right)^2 \right] dxdy, \tag{2.4}$$

where D is the image for the optical flow estimation and λ controls the relative influence of the smoothness constraint term. Iterative methods using Gauss-Seidel equations are used to find the minimum of Eq. (2.4) and estimate the optical flow

Simplified Lucas and Kanade

Simplified Lukas and Kanade OFE (LK-OFE) algorithm assumes constant or smooth motion consistency within the local window area [5], [46]. The simplified LK-OFE is an OFE algorithm using the local least-squares (LS) technique with weighting for solving a system of constraint equations [5] and can be categorized as a local optimization technique. Mathematically this can be described as follows:

$$\arg\min_{\vec{v}} \sum_{\vec{x} \in \Omega} W(\vec{x}, t)[\vec{I_s} \cdot \vec{v} + I_t(\vec{x}, t)]^2, \tag{2.5}$$

where $W(\vec{x}, t)$ is a weighting window that typically gives more influence to constraints near the center of the neighborhood Ω.

The solution for Eq. (5.14) can be described as follows:

$$\vec{v} = [A^T W A]^{-1} A^T W \vec{b}, \tag{2.6}$$

where

$$A = [\vec{I_s}(\vec{x_1}, t), \cdots, \vec{I_s}(\vec{x_n}, t)]^T \tag{2.7}$$

$$W = diag[W(\vec{x_1}, t), \cdots, W(\vec{x_n}, t)] \tag{2.8}$$

$$\vec{b} = -[\vec{I_t}(\vec{x_1}, t), \cdots, \vec{I_t}(\vec{x_n}, t)]^T \tag{2.9}$$

When $[A^T WA]^{-1}$ exists, we can solve for $\vec{v}$ using Eq. (5.18).

$$A^T WA = \begin{bmatrix} \sum WI_x^2 & \sum WI_xI_y \\ \sum WI_xI_y & \sum WI_y^2 \end{bmatrix}, \quad A^T W\vec{b} = -\begin{bmatrix} \sum WI_xI_t \\ \sum WI_yI_t \end{bmatrix} \tag{2.10}$$

Equation (2.6) is a weighted LS solution of simplified LK-OFE algorithm for optical flow.

Based on simulation results in [44] and [48], the simplified Lukas-Kanade (LK) algorithm, which is a gradient-based optical flow estimation (OFE) algorithm using the least-square (LS) technique with weighting, is one of most successful solutions in terms of noise-robustness and the number of operations.

2.2.3 Correlation-Based Optical Flow Estimation Algorithm

These algorithms define the velocity v as the shift $d = (dx, dy)$ that returns the best fit between image areas at different instances of time. This corresponds to maximizing a similarity measure over some search range.

In correlation-based optical flow estimation methods, the motion of a pixel at (x, y) in one frame relative to a successive frame is determined by a matching function, such as the sum of squared differences (SSD), sum of absolute differences (SAD), and correlation, over a searching area (SA) of $(2d+1)\times(2d+1)$ using the squared window patch of $N\times N$ pixels, which is centered at (x, y). A typical SSD matching function for OFE is

$$\underset{dx,dy}{\arg\min} \sum_{i,j} \left(I_t(x, y) - I_{t+dt}(x + dx, y + dy)\right)^2 . \tag{2.11}$$

Considering the relationship of the velocity equation,

$$velocity = \frac{\triangle distance}{\triangle time} \tag{2.12}$$

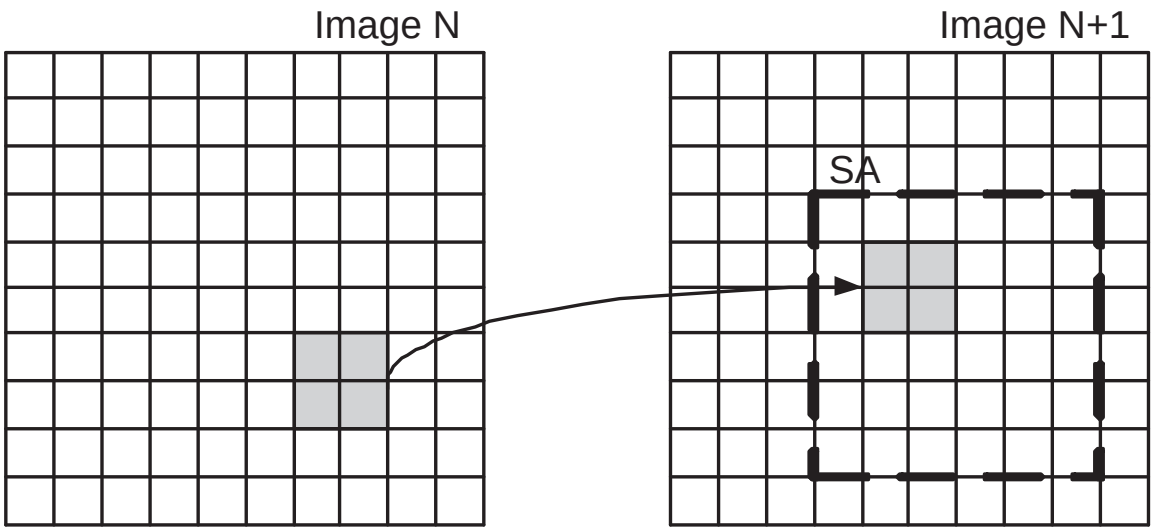


Figure 2.6. Correlation-based optical flow estimation method using spatial searching.

Correlation-based optical flow estimation methods can be categorized into spatial and temporal search methods based on the searching axis. Spatial searching methods select the best candidate pixel of the current frame within the SA of one previous frame (See Fig. 2.6). This is the case that $\triangle time = 1$ is fixed and the searching axis is spatial $\triangle distance$ in Eq. (2.12) for different velocities. The location differences between the current frame pixel (x, y) and best candidate pixel in the previous frame generate an optical flow field [2]. This requires enough memory for two frames, current and previous frames. However, the complexity is quadratic with respect to the size of the search range of μ. When $\triangle distance = 1$ is fixed and $\triangle time$ is the searching axis in Eq. (2.12), this is temporal searching case. The temporal searching method determines the best candidate pixel of the current frame over several previous frames with the fixed search range in the spatial domain (See Fig. 2.6). An optical flow field derived using a temporal searching method is calculated as $(dx/dt, dy/dt)$, where dx/dt and dy/dt are motion vectors. An advantage of this algorithm is the linear complexity with respect to the number of frames. However, the size of memory linearly increases with the number of previous frames. The Camus algorithm is a typical temporal searching method [11].

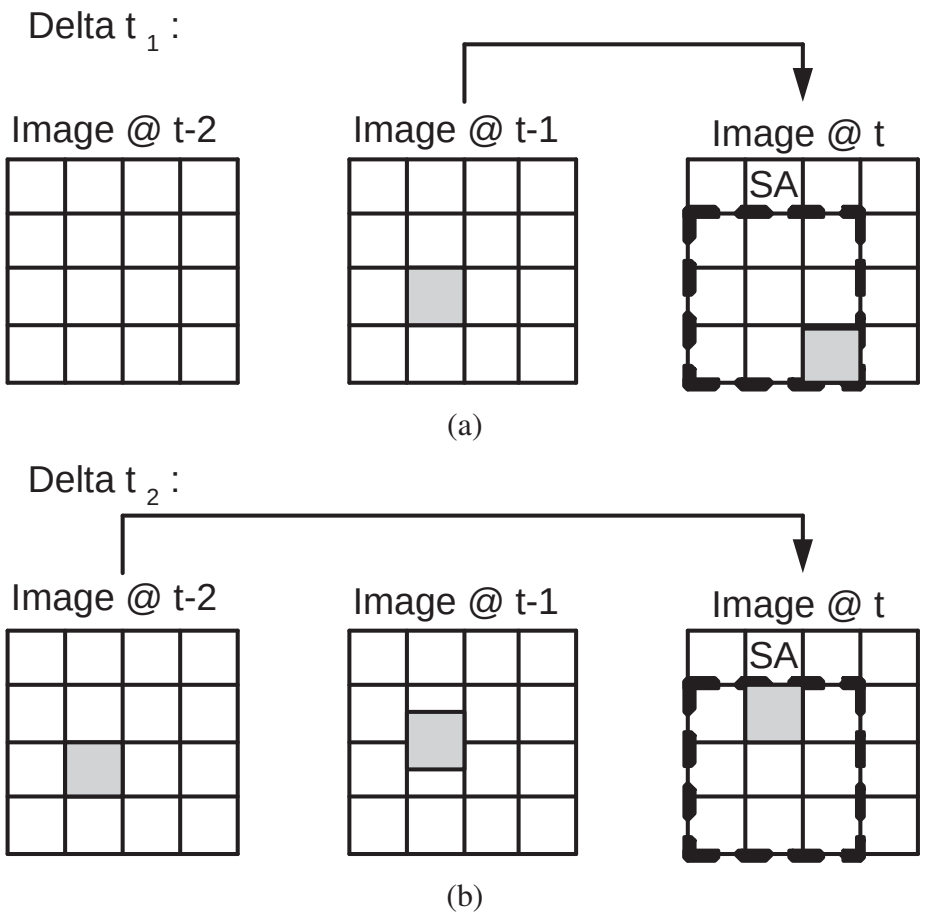


Figure 2.7. Correlation-based optical flow estimation method using temporal searching. (a) The maximum correlation case when the motion is one pixel per frame; (b) The maximum correlation case when the motion is a half pixel per frame

2.3 Motion Estimation for Video Processing

In many applications of visual communication, video signals are transmitted over relatively low transmission-rate channels compared to the amount of data in the original video signals. Therefore, video compression techniques are required to reduce the amount of video signal data. Temporal redundancy in a video signal can be reduced by motion estimation (ME) and motion compensation. The block-matching algorithm (BMA) is investigated as a motion estimation method for most video coding systems. Its goal is to find a block that is most similar to a current block within a pre-defined search area (SA) in a reference frame. As a straightforward method, the full search BMA (FSBMA) is widely investigated because

of its high performance and low control overhead [50]. FSBMA is based on a block-wise translational motion model. It can be considered as a correlation-based optical flow estimation algorithm with spatial searching. The difference between correlation-based OFE and FSBMA is that FSBMA performs a correlational spatial-searching operation per block instead of calculating motion estimation per pixel.

2.3.1 Full Search Block Matching Algorithm

The basic operation of a block-matching algorithm is to pick the best candidate image block in the reference image frame by calculating and comparing the matching functions between the current image block and all the candidate blocks inside a confined area, the so-called search area (SA), in the reference frame. The mean-of-the-absolute-difference (MAD) matching criterion is used for finding the motion vector V.

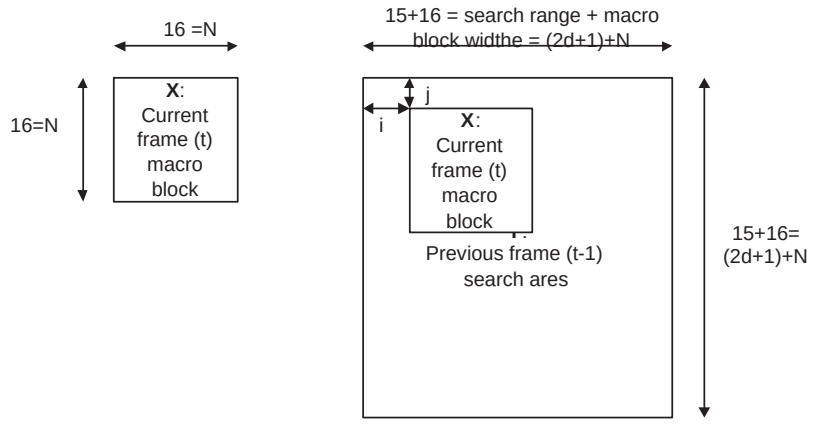


Fig. 1. The full search block matching algorithm (FSBMA) process.

Figure 2.8. The full search block-matching algorithm (FSBMA) process ($N = 16, d = 7$)**.**

Figure 2.8 illustrates the FSBMA approach. Let the present and previous frames be segmented into $N \times N$ reference blocks. FSBMA matches each block of the present frame X against each of the candidate blocks in the previous frame Y within the SA shifting from displacement $-d$ to $+d$ for i, j directions. To calculate a motion vector, we need to compare the block in the current frame with the $(2d+1)^2$ candidate block in the previous frame. And there are N^2 subtractions, N^2 absolute operations, and $N(N-1)$ additions to compute one

point of MAD. For the N values in the range of interest such as $N = 16$, $N(N-1)$ can be reasonably approximated by N^2.

$$MAD(i,j) = \sum_{m=1}^{N}\sum_{n=1}^{N} |X(m,n) - Y(m+i, n+j)|, \tag{2.13}$$

where i, j satisfy $-d \le i, j \le d$.

$$V = (i,j)|_{\min MAD(i,j)}, \tag{2.14}$$

where V is the motion vector.

The computation rate (C_r) corresponding to the number of $(2d+1)^2$ candidate blocks, the frame size $W \times H$, and the frame rate R_f is as follows:

$$C_r = (3N^2)\cdot(2d+1)^2 \cdot \Big(\frac{W\times H}{N^2}\Big)R_f = 3(2d+1)^2 \cdot (W\times H)R_f \quad [\#Operations/sec] \tag{2.15}$$

To reduce computational complexity, we can employ successive elimination technique based on triangular-inequality [40], [73].

2.3.2 Hierarchical Motion Estimation

Hierarchical block motion estimation (HBME) has been developed to reduce the computational complexity and maintain good performance compared to the full search block-matching algorithm. In hierarchical block motion estimation, the size of the block and/or the search area varies depending on the level of hierarchy. At lower levels of hierarchy, larger block sizes are used to estimate the broad motion of the image, while at higher levels of hierarchy, smaller block sizes are employed to estimate the detailed motion of the image. However, the block size in computation for motion estimation can be the same. Therefore, the HBME algorithms can be divided into two categories depending on computational block sizes at different levels, constant block size and variable block size. The three-step hierarchical search (3-SHS) is considered as one of typical constant block size algorithms. Other algorithms of this kind can be found in [53], [45], and [82]. In the variable

block size case, a promising solution is the multi-resolution motion estimation (MRME) algorithm. In conventional MRME schemes, motion vectors (MV's) are first estimated at the lowest resolution. This is reasonable since most of the image energy is preserved at this resolution. And then MV's are refined at other finer resolutions depending on the corresponding initial MV's at lower resolutions (see Fig. 2.9). Each lower resolution image is subsampled by 2 from the previous higher resolution image.

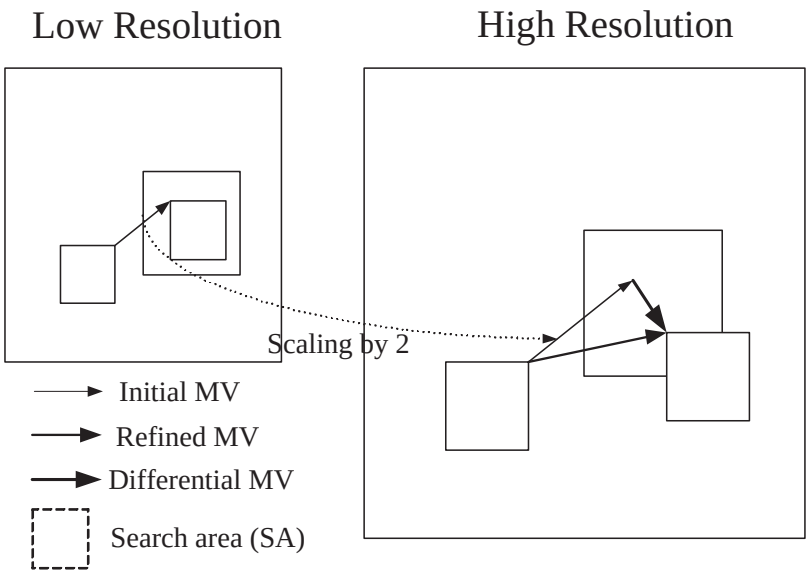


Figure 2.9. Multi-resolution motion estimation.

The computational complexity of MRME for a 16×16 block size at the finest resolution is derived below,

$$\begin{aligned} C_{mrme} &= C_{2\times2} + C_{4\times4} + C_{8\times8} + C_{16\times16} \\ &= \{K_3(w_3)^2\left(\frac{1}{2^3}\right)^2 + K_2(w_2)^2\left(\frac{1}{2^2}\right)^2 \\ &\quad + K_1(w_1)^2\left(\frac{1}{2}\right)^2 + K_0(w_0)^2(1)\,\} \\ &\quad \times (M\times 16^2)\times\left(\frac{W\cdot H}{16^2}\right)\cdot f\ (operations/s), \end{aligned} \tag{2.16}$$

where $C_{n\times n}$ is the searching complexity associated with $n\times n$ block size. $W\cdot H$ is the image

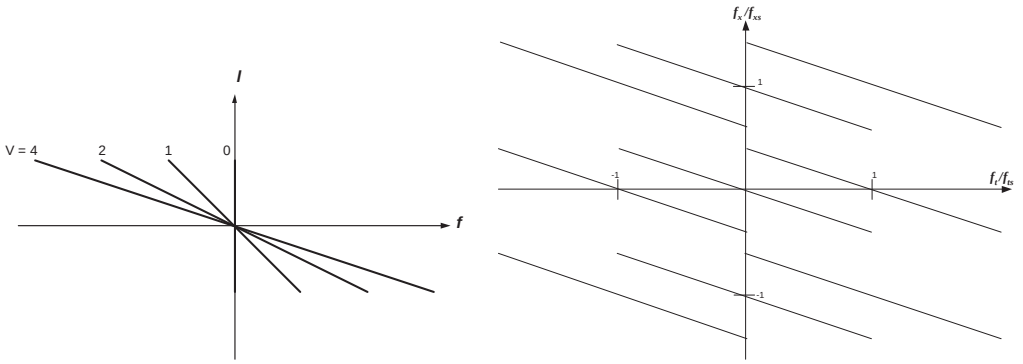


Figure 2.10. Spatio-temporal spectrum based on the constant motion model.

size, $(w_i)^2$ is the search range at level i of resolution, M is the number of operations required for finding the sum of absolute difference, f is the image frame rate, and K_i the number of images at level i of resolution. By using the proper window size, MRME algorithm can reduce computational complexity and maintain high performance.

2.4 Multi-Dimensional Filtering for Motion Estimation

Typically motion estimation algorithms assume block-wise smooth and constant motion among successive frames. This is the same assumption used for gradient-based optical flow estimation algorithms. For a one-dimensional time-varying image $i(x, t)$ moving with constant velocity v_x, this assumption can be modeled by

$$i(x, t) = i(x - v_x t). \tag{2.17}$$

After taking the Fourier transform, we obtain

$$I(f_x, f_t) = I(f_x)\delta(vf_x + f_t), \tag{2.18}$$

where $\delta(vf_x + f_t)$ is the 1-D Dirac delta function and $I(f_x)$ is the 1-D transform of the stationary image $i(x)$. Thus, it is clear that the energy of $I(f_x, f_t)$ is only on the line $f_t = -vI_x$ (see Fig. 6.3(a)).

If the image is sampled in the time and spatial domains with sampling frequencies f_t and f_x respectively, the spectrum of $I(f_x, f_t)$ in Fig. 2.10(a) will be replicated in the f_t and f_x directions at the intervals of f_{xs} and f_{ts} respectively. This is shown in Fig. 2.10(b) for a motion of more than one pixel/frame. According to Fig. 2.10, temporal aliasing will be more severe as velocity increases [12] and is hard to remove by temporal lowpass filtering as an anti-aliasing filter because the number of frames in camera device or photo sensors is fixed based on system requirements or video coding standards.

The temporal spectrum at a particular position x_o can be described by the 1-D Fourier transform:

$$I_{x_o}(f_t) = \frac{1}{|v|} \sum_{k=-\lfloor |v| \rfloor}^{\lfloor |v| \rfloor} I\left(-\frac{f_t + k}{v}\right) e^{-j2\pi(f_t - k)x_o/v}, \tag{2.19}$$

where $\lfloor |v| \rfloor$ is the largest integer less than $|v|$. The derivation of Eqn. (2.19) is described in Appendix I. This equation illustrates the temporal aliasing by the overlap of frequency responses with respect to the f_t axis when the projection of Fig. 2.10(b) into the f_t axis is happened. This aliasing can be regarded as temporal downsampling effect with linear phase modulation. The downsampling rate is determined by the speed of object and camera. Therefore, this aliasing is unavoidable for any temporal filtering operation if v is greater than 1 pixel/frame, and will be worse as v increases.

However, we can reduce the temporal aliasing phenomenon by using spatial lowpass filtering (See Fig. 2.11). This is the main reason why OFE systems need 3-D prefiltering and the idea of spatial lowpass filtering for reducing temporal aliasing can be exploited for motion estimation algorithm in video coding [38].

2.5 Low-Power System Design

In this section, we discuss about cooperative analog and digital signal processing (CADSP) approach for low-power system design and some low-power analog motion sensors will be compared. To design efficient OFE system in terms of power, size, programmability, and

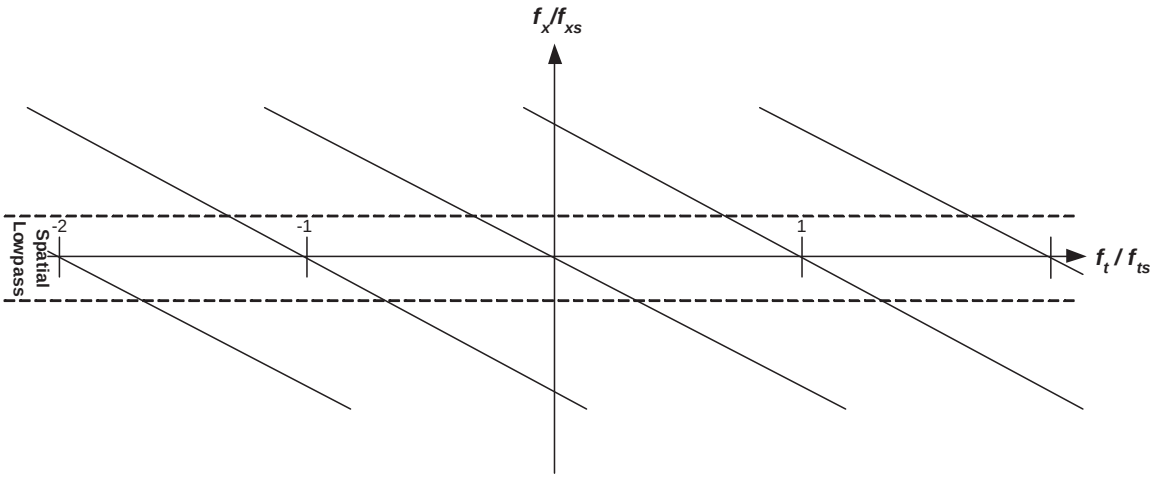


Figure 2.11. Spatial lowpass filtering range over spatial-temporal frequency domain.

performance, we suggest a concept of CADSP gradient-based OFE system.

2.5.1 Cooperative Analog and Digital Signal Processing (CADSP)

Most modern signal processing and communications occur in the digital domain, while the real world is in the analog domain. Current signal processing systems convert real-world analog signals immediately into the digital domain using an A/D converter, so most processing is implemented in the digital domain. Finally, the outputs of the digital system are reconverted to analog using a digital-to-analog (D/A) converters as a final step. One possible drawback of this typical signal processing system design is that it may consume too much power. This is undesirable for portable devices because power consumption and battery life are critical.

To alleviate the power consumption of current signal processing systems, CADSP concept is suggested in [23]. CADSP can be defined as a concept of intelligently combining programmable analog signal processing and digital signal processing techniques to achieve low-power and real-time signal processing.

Even though power consumption in DSP processors has been consistently decreasing by half about every 18 months, following Genes law [21],VLSI chips fabricated using CADSP techniques have potential to significantly surpass this improvement rate [4], [22].

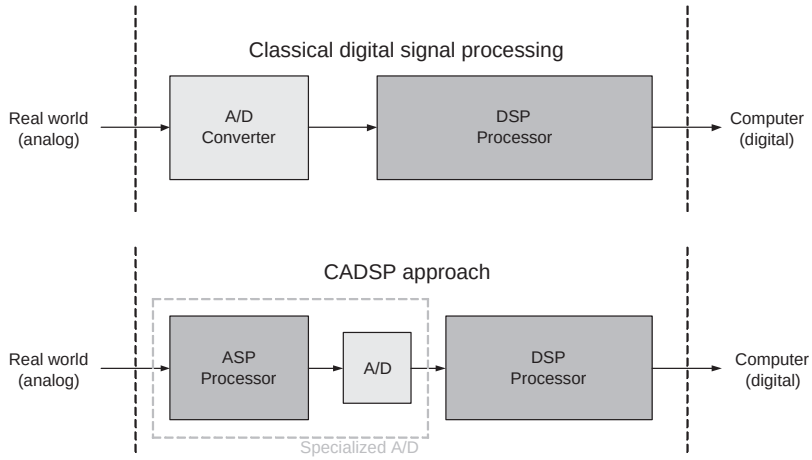


Figure 2.12. The block diagrams of classical digital signal processing and CADSP concepts [79]. In traditional DSP systems, the A/D converter is placed as close to the real-world as possible. However, significant power savings can be achieved by moving some of the signal processing functionality into the analog domain (prior to the A/D converter).

Partitioning between analog and digital computation plays a key role in a CADSP system design (Fig. 2.12). Because many analog techniques are orders of magnitude more efficient than their digital counterparts in terms of speed and power dissipation [49], we can improve power consumption efficiency by migrating processing from the digital domain into the analog domain as long as the performance can be similarly maintained. In traditional DSP systems, the A/D converter is placed as close to the real-world as possible. However, significant power savings can be achieved by moving some of the signal processing functionality into the analog domain (prior to the A/D converter). Analog floating-gate technology is often employed to achieve low-power consumption with programmability.

Although analog signal processing is capable of several important functions, the effects of bit resolution on these analog computing systems still needs more research. The computational cost typically includes various factors, such as chip area, power consumption, design time, and development cost. While the computational cost of digital computation increases linearly as the number of bits of required resolution added, the downside of the computational cost of analog computation is its exponential increase compared to that of

digital computation. Sarpeskar [60] showed that analog computation has significant advantages when the bit resolution of the input signal is no more than 10 to 12 bits.

2.5.2 Low-Power OFE System Design

Because analog processing is more economical in terms of silicon area and power consumption than digital processing of comparable complexity, some motion sensors for OFE have been implemented in analog very-large-scale-integrated (VLSI) systems [15], [28]. Etienne-Cummings, *et al.* presented a motion sensor based on spatiotemporal information and a correlation technique [15]. This shows good performance for 1-D motion detection. However, it is difficult to extend to the 2-D motion case because of the limited directional resolution and the aperture problem, which means that motion cannot be correctly decided because of the limited window size compared to the object size or moving distance. Higgins *et al.* showed correlational motion sensors based on temporal edge information [28]. Even though these motion sensors improved the performance in 2-D compared to [15], they still have some problems of aperture and limited directional resolution.

The gradient-based method is superior to the correlation-based method in terms of accuracy. Nevertheless, the latter is widely used to implement the motion sensor in analog VLSI system(s) because of limitations of bit precision and circuit noise [61]. To overcome these limitations, we can apply a CADSP approach for the motion sensor using the gradient-based optical flow estimation (OFE) method. The general goal of a CADSP approach is to assign regular and expensive computations requiring low bit resolution (less than about eight-bit resolution) to an analog processor instead of a digital processor and use a digital processor for other operations [23]. To find the best boundary of analog-digital signal processing for overall improvements in terms of power consumption, a silicon area, and cost reduction without performance deterioration,

CHAPTER 3

PERFORMANCE ANALYSIS OF SIMPLIFIED LK-OFE UNDER NOISY ENVIRONMENTS AND RAPID ALGORITHM VERIFICATION

Considering the performance of a CADSP system, analog domain noise factors must be considered in addition to purely digital domain processing effects. To incorporate analog and digital domain noise factors, signal modeling for CMOS image sensors is described in section 3.1. Experimental results based on the signal modeling are presented in section 3.2. Rapid algorithm verification based on CADSP transform imager and simulator is explained in section 3.3. Finally conclusions are described in section 3.4.

3.1 Signal Modeling for CADSP Imaging Systems

In CMOS circuits, there are some typical circuit noise factors such as thermal, shot, and flicker noises (intrinsic noise factors). Even though other noise factors exist such as power supply fluctuation, electromagnetic fields, and *etc.*, those factors can be easily suppressed. However, intrinsic noise factors can only be reduced by an improved circuit design and optimization practice. Thermal noise is generated by random thermally induced motion of electrons inside an electrical conductor at equilibrium. Shot noise consists of random fluctuations of the electric current when carriers cross a depletion region. This is caused by the fact that the current is carried by discrete charges (electrons). Flicker noise is caused by traps due to crystal defects and contaminants in electric devices and is also well-known as pink noise, or $1/f$ noise. If the frequency of CMOS circuit is fast enough and operating below threshold, then the dominant noise factor of CMOS image sensors becomes shot noise [68]. The intrinsic noise factors that affect CMOS imagers, have a direct impact on the overall system performance of a CADSP system that incorporates a CMOS imager. This leads to test behavioral-level simulations to evaluate the performance.

In order to design efficient gradient-based OFE system (Fig. 2.5) using CADSP approach, some behavioral-level simulations are needed to be executed for validating or evaluating which stage of a gradient-based OFE algorithm we have to partition for the analog and/or digital domains. Another reason behavioral-level simulation is to estimate the performance of algorithms under noisy environments. These simulations must include the effects of the digital and analog circuit noise factors of a CADSP approach. The simplified signal model for simulations is as follows:

$$y(n_1, n_2) = (1 + p(n_1, n_2))x(n_1, n_2) + q(n_1, n_2), \tag{3.1}$$

where

$$\begin{aligned} p(n_1, n_2) &= r(n_1) + c(n_2) + e(n_1, n_2) \\ q(n_1, n_2) &= a(n_1, n_2) + b(n_1, n_2) \end{aligned}$$

In the above equations, $y(\cdot)$ is a noisy signal, $p(\cdot)$ is a multiplicative noise, $x(\cdot)$ is the original signal, and $q(\cdot)$ is a additive noise. In Eq. (3.1), $p(\cdot)$ is mainly due to fixed pattern noise (FPN), a source of noise in the CMOS image sensor, such as row, column, and element-wise factors and $q(.)$ is composed of additive Gaussian noise (AGN) by shot and thermal noise factors and bit resolution/quantization noise by analog-digital converter (ADC) bit resolution capability and/or linear range of analog circuitry.

Using the signal model in Eq. 3.1, several simulations were executed to find the performance effects of each of the noise components. First, we performed bit-level simulations of stage (i) in section. 2.2.2 to calculate the effects of bit-solution noise; then, to consider other additive noise of an analog processor, we tested the additive white Gaussian noise (AWGN) model; Under the assumption that FPN or multiplicative noise (MN) is small, we can be model it as AGN, additive noise case simulations are good approximations for our signal modeling. If MN needs to be considered, FPN factor should be included for our simulations.

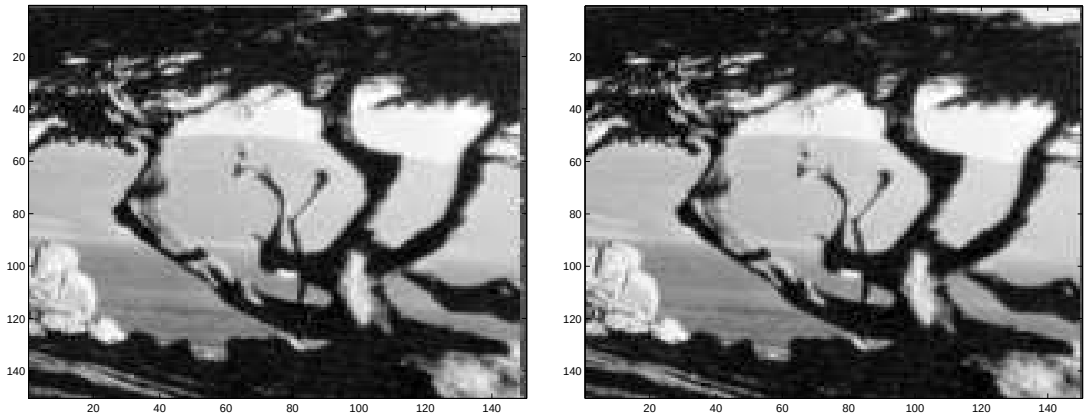


Figure 3.1. A part of sample sequence of translating tree images for simulations. (a) 19th image; (b) 20th image

3.2 Simulation and Discussion

In this section, we show the simulation results of the simplified LK-OFE, LS-OFE, method resulting from bit resolution and noise power. The effect of each noise component in Eq. (3.1) is tested by exciting only one noise source at a time. The code for the simplified LK-OFE method is a modified version of the code supplied at [5]. A sequence of translating tree images was used as a sample sequence. Figure 5.4 shows part of these sample images. For the derivative filter, we used $\frac{1}{12}(-1, 8, 0, -8, 1)$.

3.2.1 The Bit Resolution Noise (BRN) Case

This section discusses the bit-level simulation for the smoothing step in the simplified LK-OFE, LS-OFE, method. The simulation was divided into two categories. For the first category, the same bit resolution (constant bit resolution case) is used for all three-dimensional input signals. For the second category, different bit resolutions (different bit resolution case) are used for input signals. The bit resolutions we used for the coefficient ranged from five to eight bits. Three-dimensional (3-D) smoothing was performed with a spatiotemporal Gaussian filter of standard deviation of 2.0 pixels per frame. One advantage of using Gaussian kernel is that the summation of kernel coefficients is equal to 1 because

it is a probabilistic density function (pdf). Therefore, the range of values are not changed after filtering except the fractional bit resolution.

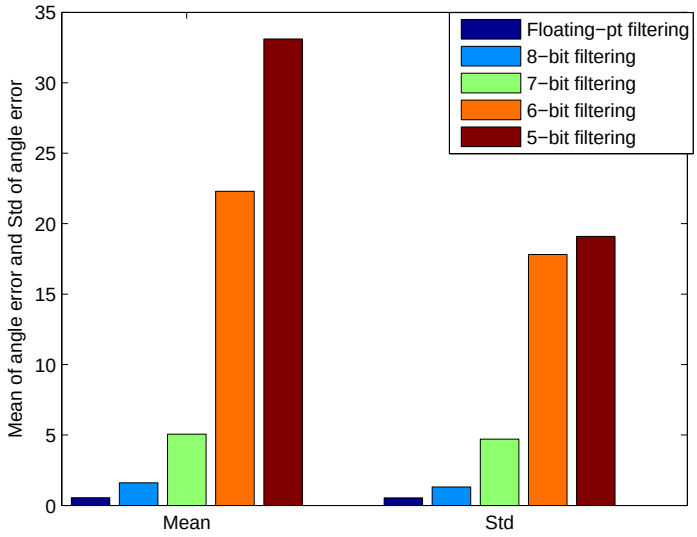


Figure 3.2. Simulation results of mean and standard deviation of angle error for different signal bit resolutions of all three-dimensions. The coefficient is eight bit-resolution.

Figure 3.2 shows the angle error performance of LS-OFE based on different signal bit resolution with 8-bit coefficient values. Figure 3.3 show the angle error performance of LS-OFE depending on different signal and coefficient bit-resolutions. For the constant bit resolution case, the reduced bit resolution of the filter coefficient only slightly affects the performance up to around five bits, but the performance greatly deteriorates as the bit resolution of the input signal reduces further (See Fig. 3.2 and Fig. 3.3). Even though the performance of eight-bit resolution is very little different compared to that of full resolution Fig. 3.2, the mean of angle error (MAE) exponentially increases as the bit solution of the input signal reduces below eight bits. The performance results are presented in Fig. 3.3.

The performance deterioration can be improved for the different bit resolution cases.

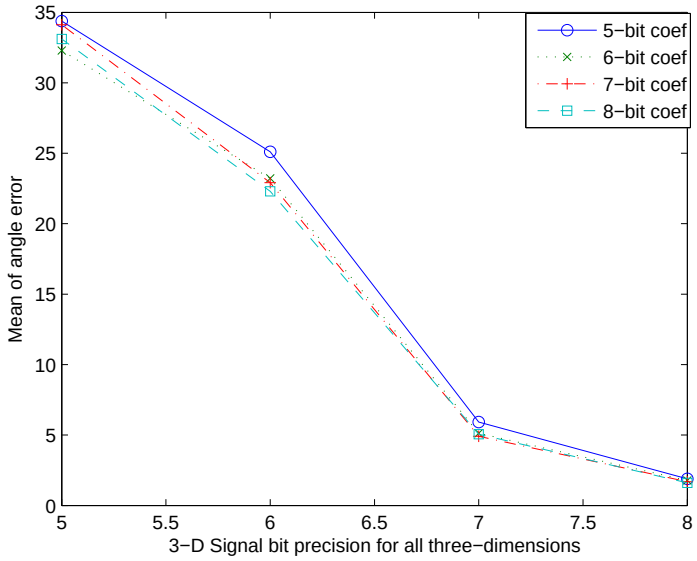


Figure 3.3. Simulation results of mean of angle error for different signal bit resolutions of all three-dimensions.

If the reduced bit resolution is only applied to the steps of vertical and/or horizontal filtering(s) in three-dimensional smoothing with eight-bit resolution for other dimensional smoothing results, the performance can be improved depending on the direction of motion activity (see Fig. 3.5 and Fig. 3.6). This phenomenon is self-explanatory if we consider the motion activity of sample test sequence, which is purely horizontal translational movement. Since our 3-D FIR filtering is separable filtering in terms of horizontal, vertical, and temporal directions, the performance would be much more affected by the same directional filtering steps. Therefore, Fig. 3.4 and Fig. 3.5 have almost same performance and Fig. 3.6 has very little performance drop. This phenomenon can be easily proved by considering 3-D Fourier Transform relationship, which is explained in section. 2.4 . The more interested readers can find more materials in [67].

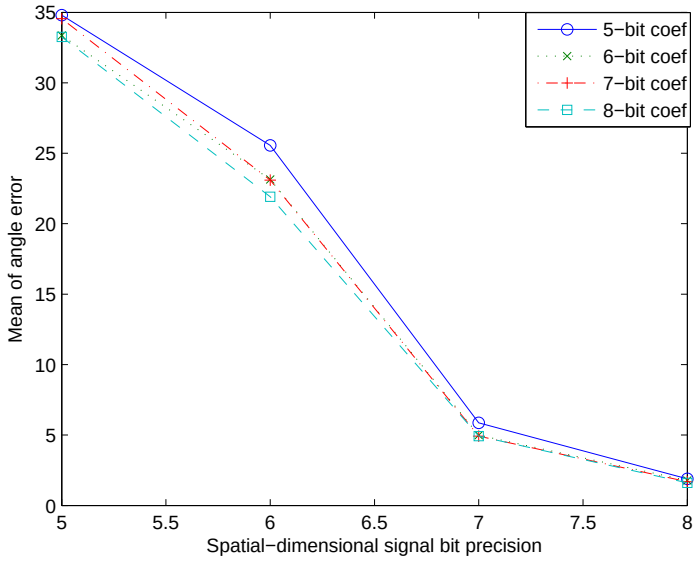


Figure 3.4. Simulation results of mean of angle error for different signal bit resolutions of spatial dimension.

3.2.2 The Additive Gaussian Noise (AGN) Case

Besides bit resolution noise resulting from the limited linear range in the circuitry, other noises exist such as shot, thermal, *etc*. Generally, the thermal and shot noises can be modeled as additive Gaussian noise (AGN). Therefore, we need to include the AGN model for behavioral simulation. It is also a good approximation for other general noises since most random variables can be modeled as a normal distribution with the help of central limit theorem (CLT). For normalized filter coefficients, the linear combination of Gaussian noise terms is another Gaussian noise under i.i.d (independent identical distribution). From this viewpoint, the AGN model is reasonable for a linear filter. Figure 3.7 shows the results of the LS-OFE algorithm when AGN is used for Eq. 3.1. As we expected, the MAE increases as noise level goes up for the filter. If we compare the results of the BRN and the AGN cases, the performance of AGN is better than those of BRN with similar or same variance cases. One reason is that BRN has a uniform distribution and AGN follows normal distribution model. It means that BRN typically contributes more noise component than AGN for performance. Another reason is that LS-OFE is the optimal solution for AGN. Therefore,

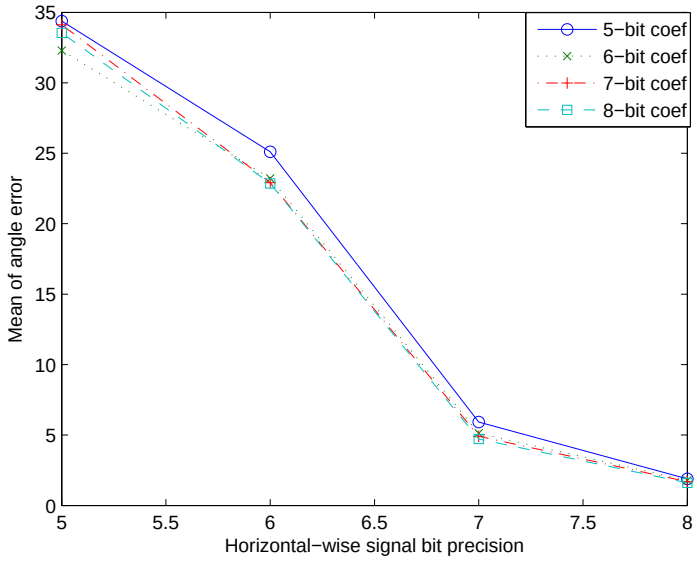


Figure 3.5. Simulation results of mean of angle error for different signal bit resolutions of horizontal dimension.

simplified LK-OFE can maintain good performance under AGN environments.

3.2.3 The BRN and AGN Case

In this section, we show the test result of the combined additive noise case using BRN and AGN models. Figure 3.8 shows that the differences of the MAEs for various signal bit-resolutions under different degrees of normal distribution noise. The equation of signal-to-noise ratio (SNR) for the BRN and AGN models can be described like Eq. (3.2).

$$SNR_{brn+agn} \propto \quad log_{10}\Big(\frac{\sigma_x^2}{K \times 2^{-2b}(R_{FS})^2 + \sigma_{agn}^2}\Big), \tag{3.2}$$

where K is a constant, b is the number of bit resolution, FS is the full linear range, σ_x^2 is signal power, and σ_{agn}^2 is noise power for BRN and AGN modeling.

When the bit resolution noise is large, such as five or six-bit for signal bit resolution, then the AGN sources, which are smaller than the step sizes of the bit resolutions, can be

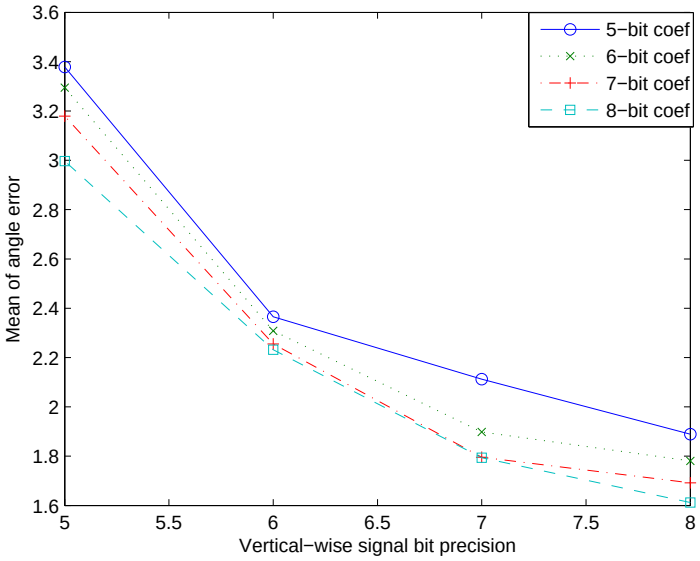


Figure 3.6. Simulation results of mean of angle error for different signal bit resolutions of vertical dimension.

ignored because of the large step size. In such cases, the effect on SNR of ignoring the AGN sources is negligible. This can be seen by noticing that the differences between the results of Fig. 3.8 and Fig. 3.4 around five and six-bit signal resolution cases. When the bit resolution is close to full resolution, such as seven or eight-bit for signal bit resolution, then AGN can affect the performance because the bit resolution noise is also similar level in Eq. 3.2. Therefore, the total noise is mainly dependent on BRN for signals of low bit resolution cases. For other cases, AGN and BRN must should be considered both. As a result, the total noise model is dependent on the ratio of noise levels of the BRN or AGN models.

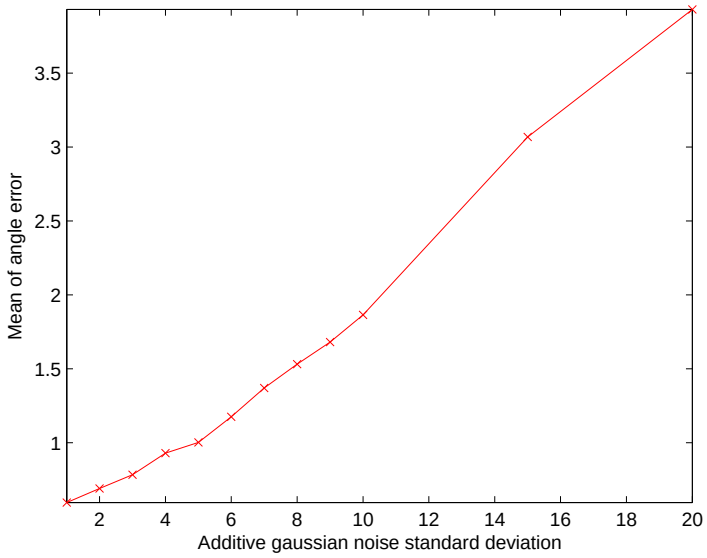


Figure 3.7. Simulation results of mean of angle error of LS-OFE with AGN for different standard deviations.

3.2.4 The Multiplicative Noise (MN) Case

If we need to model other noise factors in addition to additive noise factors, multiplicative noise components must be considered. The gain variation of image sensors in imager causes multiplicative noise, which appears in the image as fixed pattern noise (FPN) [13], [9], and [10]. Since each image sensor has a pixel-wise amplifier/buffer, generally pixel-wise FPN exists. Depending on the architecture of imager, there are also row and/or column-wise FPN. Typically CMOS imager employs column amplifier/buffer (See Fig. 2.2). Thus, it generates column-wise FPN, which is a dominant multiplicative noise factor in Eq. 3.1. We test the performance of MN for column-wise FPN under Gaussian distributions. For the Gaussian distributions, the MAE for any FPN case deteriorates more as the scaling percentage of Normal distributions increases, which is the standard deviation (std) for MN factor (See Fig. 3.9). Even though the std is very small, the performance can be easily affected because the effective noise is dependent on signal magnitude. However, the trend of performance can be different depending on the direction of motion activity and photo-detector access in an imager. For example, the performance

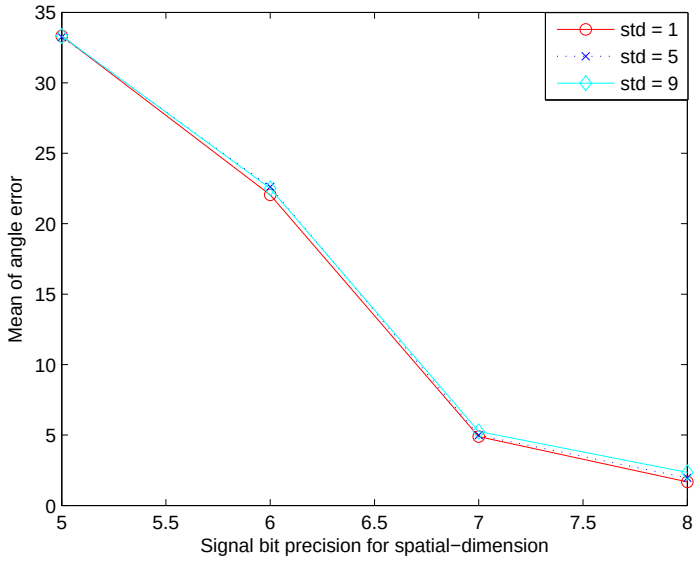


Figure 3.8. Simulation results of mean of angle error for different signal bit resolutions of spatial dimension with different AGN.

due to row-wise FPN is much better than that due to column-wise FPN for horizontal translational moving image sequences. The performance of element-wise Gaussian distributed MN is also tested (See Fig. 3.10). The performance degradation due to element-wise MN is less than that of column-wise MN. In addition, element-wise MN has not directional sensitivity since it is spatially isotropic.

3.2.5 The BRN, AGN, and MN Case

For a very general case, we test the simplified LK-OFE method under BRN, AGN, and MN models. This simulation model considers the most realistic imager situation for the optical flow system according to Eq.(1). The result is presented in Fig. 3.11 and Fig. 3.12. Considering the same percentage of additive and multiplicative noise cases with normal distribution, multiplicative column-wise FPN has a greater influence on performance than AGN. However, signal bit resolution has a greater effect than any other noise factors under noise levels in Fig. 3.11 and Fig. 3.12.

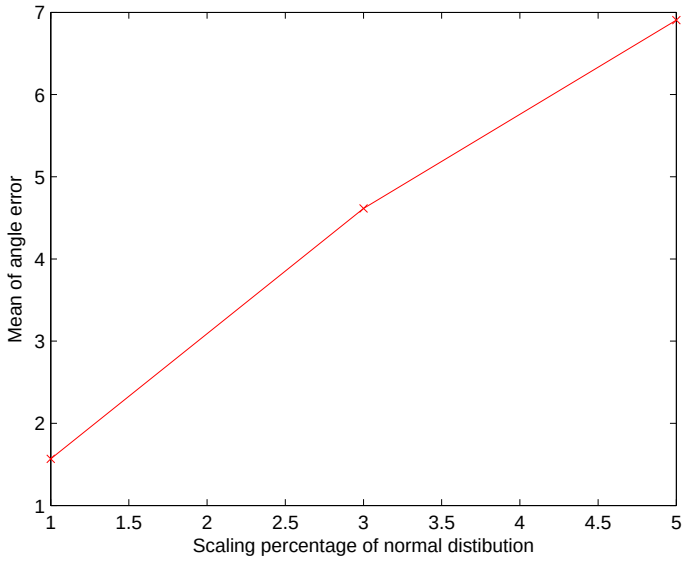


Figure 3.9. Simulation results of mean of angle error for different percentages of normal distribution for column-wise FPN.

3.3 Rapid Algorithm Verification for Cooperative Analog-Digital Imaging Systems

Typically the development of an analog system requires a lengthy process relative to digital systems if the system complexity is similar. To expedite the design cycle of CADSP system, it is advantageous to design and test the algorithm while the hardware is being developed. For this purpose, a simulation tool is developed based on our signal modeling 3.1. If the candidate analog system is undetermined, purely algorithm-level simulation with noise modeling is best-effort. However, better performance estimation can be obtained by incorporating system architecture modeling if the candidate systems are already fixed. Therefore, this simulator must take into account both the architectural attributes as well as the physical noise characteristics [39].

Our algorithm verification using simulator has dual-purpose. First, it helps to predict the algorithm performance before the completion of analog and digital hardware implementation. Second, noise tolerance can be reported to analog design team; this ensures that the analog system meets expected performance.

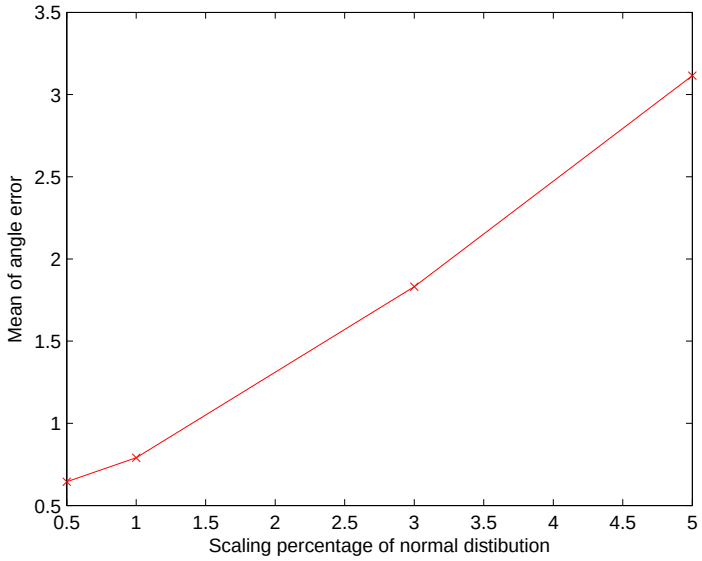


Figure 3.10. Simulation results of mean of angle error for different percentages of normal distribution for element-wise FPN.

In following sections we will describe about a separable transform imager (STI) for a CADSP imaging system and imager simulator based on the STI and our signal modeling.

3.3.1 Separable Transform Imager (STI)

The transform imager was designed to allow for high-level bio-inspired computation in a programmable architecture that still possesses pixels with high fill-factor like those in active pixel sensor (APS) imagers [25]. The block diagram of the transform imager is shown in Fig. 3.13. Each pixel outputs a current proportional to a multiplication of an input voltage and the photo-sensor current. When these pixels are configured in a matrix architecture, where the inputs are column broadcast voltages and output currents are summed along orthogonal rows, a matrix-vector computation is obtained on the pixel array. This system utilizes floating-gate circuits to store the transform bases and compute matrix-vector computations [24].

The imager array is designed such that multiplications can be performed at the pixel level. In essence, each pixel consists of a differential pair connected to a photodiode that

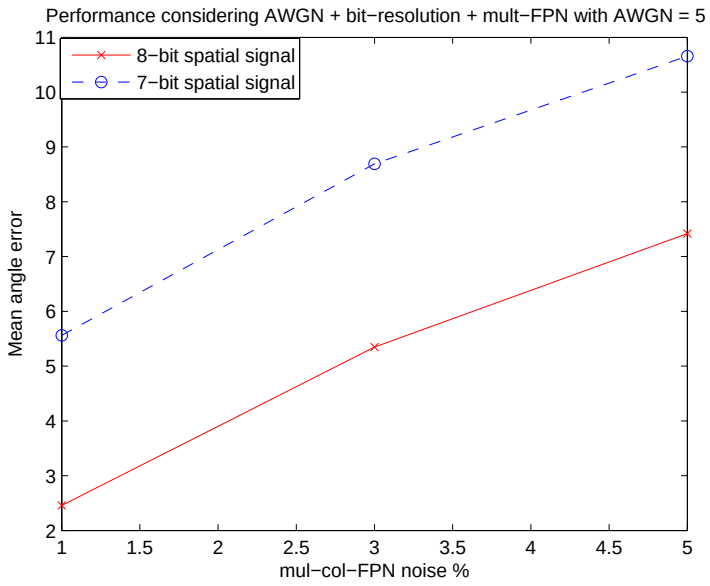


Figure 3.11. Simulation results of mean of angle error for different signal bit resolutions of spatial dimension and column-wise MN with sdt = 5 for AWGN.

acts as a current source (Fig. 3.14). The input voltage V_+ and V_- are provided by the floating gate elements, which are pre-programmed with coefficients. The reference current I_{photo} is produced by light in the scene. The output currents I_+ and I_- are then taken differentially, and, when operating in sub-threshold region, they are given by [3]

$$I_+ - I_- = I_{photo} \tanh\left(\kappa \frac{V_+ - V_-}{2U_T}\right),$$

where κ is a device parameter and U_T is the temperature-dependent thermal voltage. Since the hyperbolic tangent function behaves like a linear function for small arguments, the differential current can be expressed as

$$I_+ - I_- = I_{photo}\,(V_+ - V_-)\left(\frac{\kappa}{2U_T}\right)$$

for small $(V_+ - V_-)$. κ is a main source for MN. Based on [3], this is about 1% as element-wise FPN.

In this imager, a separable matrix transform can be performed as

$$\mathbf{Y} = \mathbf{A}^T\mathbf{P}\mathbf{B}\ ,$$

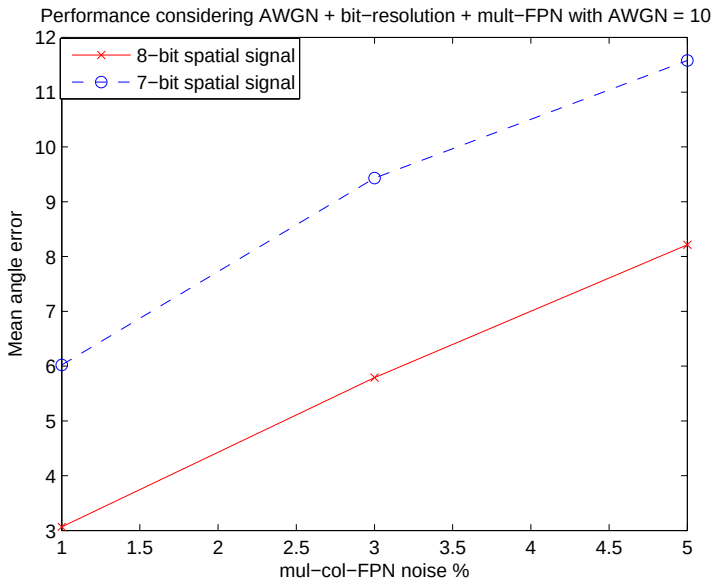


Figure 3.12. Simulation results of mean of angle error for different signal bit resolutions of spatial dimension and column-wise MN with sdt = 10 for AWGN.

where **P** is the imager pixel array, **Y** is the computed output image array, and **A** and **B** are the transform basis functions.

3.3.2 Imager Simulator

An imager has to communicate with subsequent processing at some point. In the case of the transform imager, it communicates with a host computer through a simple application programming interface (API). The imager simulator is designed to have the same API as the transform imager. This modular approach makes co-development of hardware and software possible. As depicted in Fig. 3.15, algorithms under development can be run and verified in both the transform imager and imager simulator using a common set of API.

In terms of implementation, the imager simulator is divided into two levels: behavioral and physical simulation. Behavioral simulation captures the computational behavior of the transform imager; it assumes ideal physical characteristics of the CMOS circuits. Algorithms that take advantages of the focal-plane processing capabilities can be quickly tested for feasibility. On the other hand, the physical simulation embodies the non-idealities of

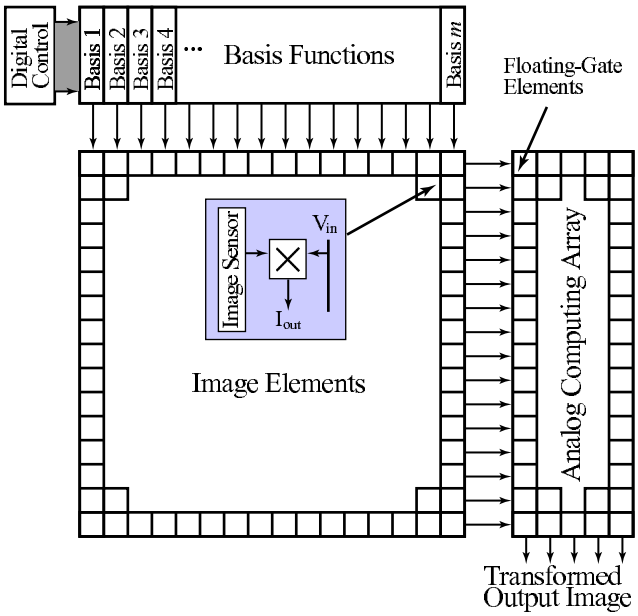


Figure 3.13. Top view of the CADSP matrix transform imager [25], [24]. This architecture allows for arbitrary separable matrix image transforms; these transforms are programmable because floating–gate circuits are used to store transform bases. Voltage inputs from various bases are broadcast along columns, and output currents are summed along lines on each row. Each pixel processor multiplies the incoming input with the measured image sensor result, and outputs a current of this result. Basis functions could be from spatial oscillators, pattern generating circuits, or arrays of floating-gate storage elements.

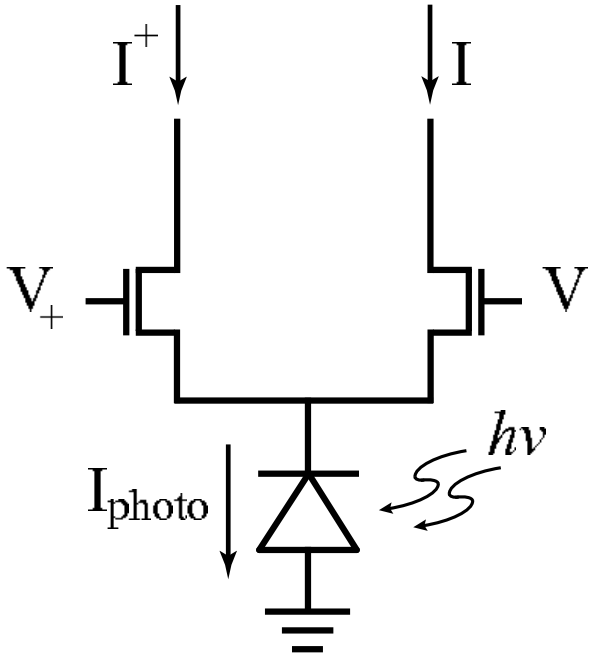


Figure 3.14. Transform imager sensing element (pixel). A differential pair is used to modulate the fraction of the sensor current; this effectively multiplies the transduced photodiode current by incoming basis functions (from the differential voltages). For sufficiently small differential input voltages, a linear multiplication is obtained.

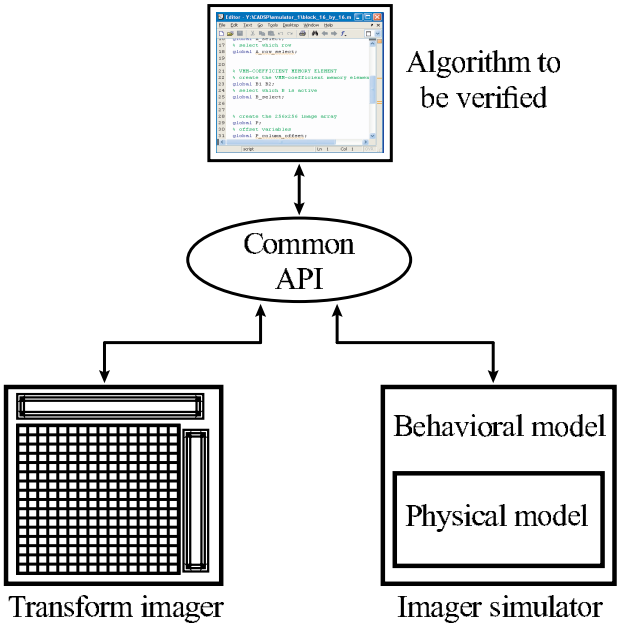


Figure 3.15. CADSP imager and MATLAB-based simulator API structure. Both the imager and simulator share the same API. Algorithms designed for the imager ca be verified in the simulator through this common API.

the transform imager, such as device mismatches and photodiode dark current. Imager designer and algorithm developers can use the physical simulation to model how the actual imager behaves.

Non-ideal behaviors manifest in different parts of the imager. Two major non-idealities are noise and device mismatch [57]. To model these physical characteristics, a MATLAB "pixel" object is defined to incorporate such parameters as differential pair mismatch, additive noise, and multiplicative noise. A "pixel-column" object is then defined with the "pixel" objects and column noise parameters. These "pixel-column" objects form an "image-plane" object, which, in turn, is a major component of the "imager" object. The behavioral simulator builds on this physical model; it follows the programming and image-capture procedures as one would control the imager hardware.

3.3.3 Case Study: Gradient-Based Optical Flow Estimation System

To develop a gradient-based optical flow estimation system in a CADSP imaging system, we need to partition the algorithm into digital and analog parts. Typically, gradient-based

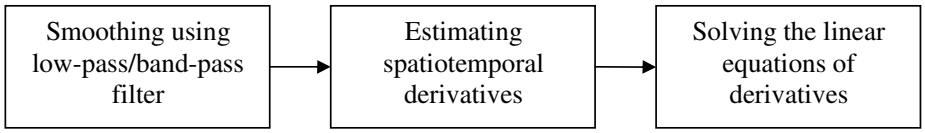
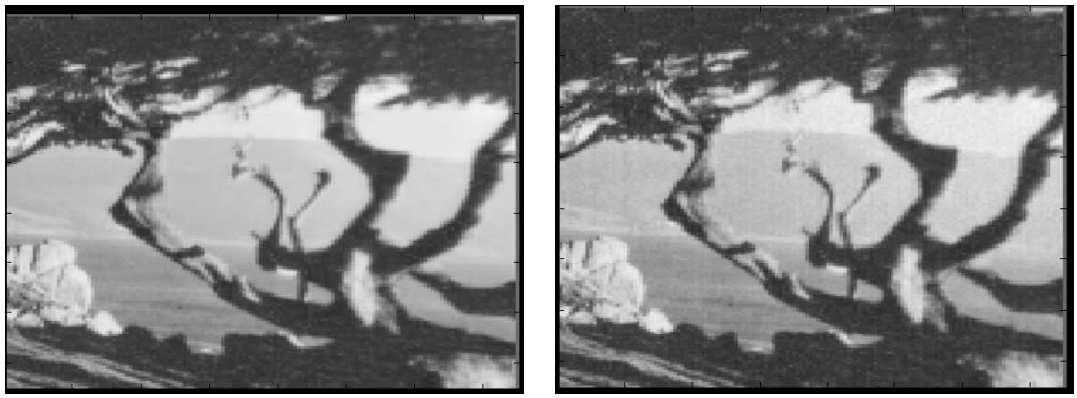


Figure 3.16. Block diagram of gradient-based OFE algorithms. In our CADSP algorithm simulation, the first two filtering blocks with respect to spatial domain are mapped to the transformed imager simulator.



(a) (b)

Figure 3.17. The original and noisy sample images using noise modeling. (a) Original translational sample image; (b) The sample image of (a) with noise modeling

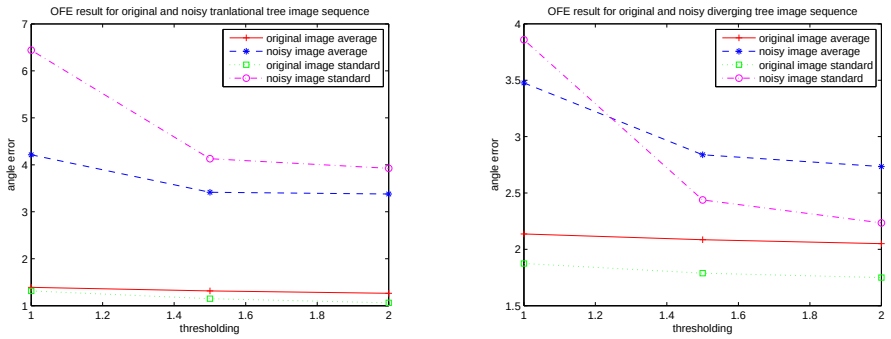
optical flow estimation methods can be divided into three stages:

(i) Pre-filtering or smoothing with low-pass/band-pass filters that enhance the signal-to-noise ratio (SNR).

(ii) Extracting spatio-temporal derivatives.

(iii) Producing a two-dimensional (2-D) optical flow field by solving the linear equations of derivatives (see Fig. 3.16).

In comparison with stage (iii), filtering stages (i) and (ii) are not affected much by bit resolution if bit resolution is higher than 6 bits [35]. Therefore, filtering stages are good candidates for analog domain processing. In stage (iii) of Fig. 3.16, gradient-based OFE methods can employ local optimization methods based on a set of constraint equations [5].



(a) (b)

Figure 3.18. The performance results of gradient-based OFE. In these figures, "standard" means standard deviation. (a) Comparison of OFE performance between original and noisy translational tree image sequence; (b) Comparison of OFE performance between original and noisy diverging tree image sequence.

Since the spatial filtering of the first two stages (in Fig. 3.16) are mapped into analog domain processing, these processing can be performed in the transformed imager. An efficient approach for implementing the filtering steps is the checkerboard-type filtering (CBTF) [36]. Weighted least-squares optimization method is employed for stage (iii). However, any low-complexity method can be used in stage (iii).

Various noise terms in the analog domain (e.g., additive and multiplicative noise) are incorporated for algorithm verification. Figure 3.17 shows an original image and a noisy image after adding noise values. Column-wise and additive noise are self-evident. For two test video signals, diverging and translational tree, algorithm verification is performed with our noise modeling. The performance results are presented in Fig. 3.18. The eigenvalue thresholding is applied to improve performance. Usually, the optical flow density of our CADSP system is higher than that of the digital processing systems. As expected, the performance of the CADSP approach is slightly less accurate than that of the digital processing systems. However, the CADSP approach is beneficial if the system cost metric for power consumption or computational complexity is more important than a small penalty in accuracy. Since the performance of CADSP system degrades with the increase of noise levels, constraints of noise level in analog system should be stipulated.

Therefore, our algorithm verification has dual-purpose. It helps to predict the algorithm performance before the completion of analog and digital hardware implementation. In addition, noise tolerance can be reported to analog design team; this ensures that the analog system meets expected performance.

3.4 Conclusions

In this chapter, we presents our signal modeling to consider circuitry noise factors for algorithm performance tests. For a better simulation, not only additive noise factors but also multiplicative noise factors should be considered. Based on our simulation, LS-OFE shows good performance for AGN case. However, LS-OFE shows less accurate performance for BRN. This is reasonable since LS is the best estimator under additive Gaussian noise. If 8-bit resolution is employed for filtering, performance degradation due to BRN can be neglected. For multiplicative noise factors, the performance can be easily affected by the small multiplicative noise factors. If reader is interested in the performance analysis of correlational method OFE algorithm under noisy environments, you can refer [37].

Based on our signal modeling, we developed a simulator to achieve rapid algorithm verification. This makes us to develop and verify algorithm without waiting for the hardware full development. It helps to predict the algorithm performance before the completion of analog and digital hardware implementation. In addition, noise tolerance can be reported to analog design team; this ensures that the analog system meets expected performance.

CHAPTER 4

FILTERING ALGORITHM FOR OFE SYSTEM WITH FOCAL PLANE IMAGER

In this chapter, we describe a new filtering algorithm for prefiltering and derivative operations in gradient-based OFE algorithms using a transform imager, which is capable of analog domain processing [58]. Previous work regarding filtering steps in a gradient-based method are reported in [63] and [18]. Some papers about derivative filter kernels are presented in [17] and [16]. However, most of the methods focus on filtering accuracy and real-time issues. This chapter is divided into four sections. Basic computational procedure of the separable transform imager is illustrated in section 4.1. Mathematical equivalent processing model of filtering steps is introduced in section 4.2. Checkerboard-type filtering is explained in section 4.3. Comparison and discussion are shown in section 4.4. Finally conclusions are described in section 4.5.

4.1 Basic Computational Procedure of the Separable Transform Imager (STI)

Members of the CADSP group are developing a separable transform imager (STI), which is a programmable analog transform imager [58]. The STI is implemented using floating-gate technology, which operates in the subthreshold region to reduce power consumption. The goal of the STI is to support the basic spatial filtering for image/video processing within imager in analog domain processing. The target functionalities include separable row/column-wise directional global/block FIR filterings and separable row/column-wise directional block transforms.

In many image/video processing applications, the fundamental operations are 1-D/2-D FIR filterings and 1-D/2-D block transforms. Mathematically these operations are based on dot products in matrix operations. The basic linear filtering operation - essentially a convolution - can be written in a matrix form. For example, the 1-D convolution $y[n]$ =

$x[n] * h[n]$ can be written as

$$\vec{y} = \mathbf{A}^T \vec{x}, \tag{4.1}$$

where the vectors $\vec{x}$ and $\vec{y}$ contains the entries of $x[n]$ and $y[n]$, respectively, and the transform matrix $\mathbf{A}^T$ contains shifted versions of the impulse response $h[n]$ for an N-point vertical directional 1-D FIR filter:

$$\mathbf{A^T} = \begin{bmatrix} h[0] & 0 & \dots & 0 & 0 \\ h[1] & h[0] & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & h[N-1] & h[N-2] \\ 0 & 0 & \dots & 0 & h[N-1] \end{bmatrix} \tag{4.2}$$

where the superscript T represents transpose (assuming all matrix coefficients are real). Equation 4.2 is called the Toeplitz matrix in linear algebra. Vector forms such as $x[n]$ and $y[n]$ in Eq. 4.1 are good for 1-D signal input and output. To model image date, we need 2-D array form in a matrix such as $\mathbf{X} = [\vec{x_1} \quad \vec{x_2} \quad \dots \quad \vec{x_n}]$. To perform vertical directional 1-D FIR filtering for 2-D image data, a Toepolitz matrix in Eq. 4.2 can be applied to individual columns of $\mathbf{X}$. The result of this vertical directional 1-D FIR filtering is $\mathbf{Y} = [\vec{y_1} \quad \vec{y_2} \quad \dots \quad \vec{y_n}]$.

For 2-D image data case, Eq. 4.1 can be expressed as

$$\mathbf{Y} = \mathbf{A^T X} \tag{4.3}$$

For horizontal directional 1-D FIR filtering, the filtering in Eq. 4.2 is also applied to the rows of $\mathbf{X}$. Assume that $\mathbf{B}$ is another Toeplitz matrix for horizontal directional 1-D FIR filtering. Convolution with rows of $\mathbf{X}$ can be expressed as

$$\mathbf{Y} = \mathbf{XB}, \tag{4.4}$$

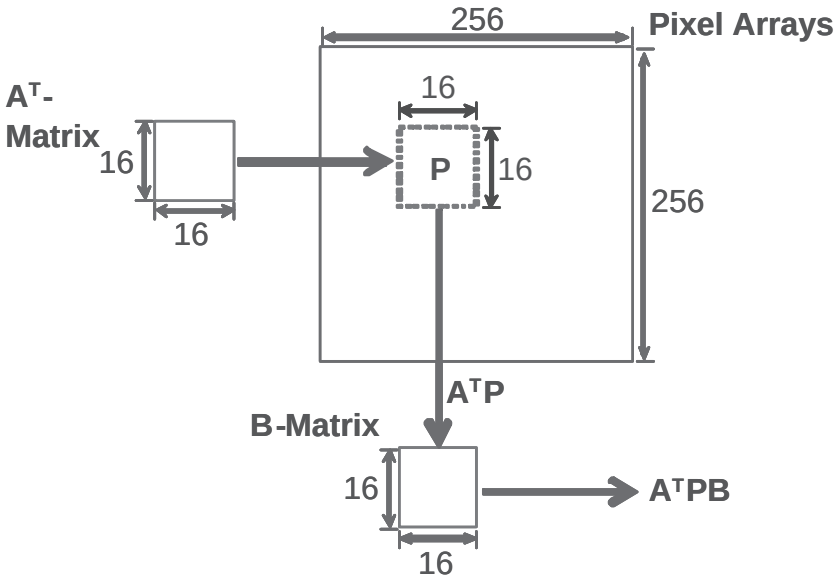


Figure 4.1. Imager architecture to perform matrix multiplication.

where **X** and **Y** represent input and output images, respectively. Therefore, combining Eq. 4.3 and Eq. 4.4, a general separable 2-D FIR filtering/transform can be written as

$$\mathbf{Y} = \mathbf{A^T P B}, \tag{4.5}$$

where P represents 2-D pixels arrays on the imager for an input image. The separable transform imager is designed to perform Eq. 4.5 on the focal plane - the 1-D/2-D FIR filtering or 1-D/2-D block transforms are computed on the pixels while an image is being captured [58]. A high-level view of its computational procedure is shown in Fig. 4.1. The imager features a 256×256 pixel array. The matrix **A** and **B** are stored (programmed) in separate analog memory elements, both of size, 16×16. The matrix multiplication $\mathbf{A}^T\mathbf{PB}$ is computed on 16×16-blocks **P** on the pixel array. Figure. 4.2 shows a detailed view of the matrix multiplication. First, the dot products of the first row of **A** and all the columns of **P** are computed. The intermediate dot products are then multiplied with the rows of **B** (again, dot product). The resulting vector is the first column of the result **Y**. This process is then repeated on every row in **A** to compute the remaining columns of **Y**.

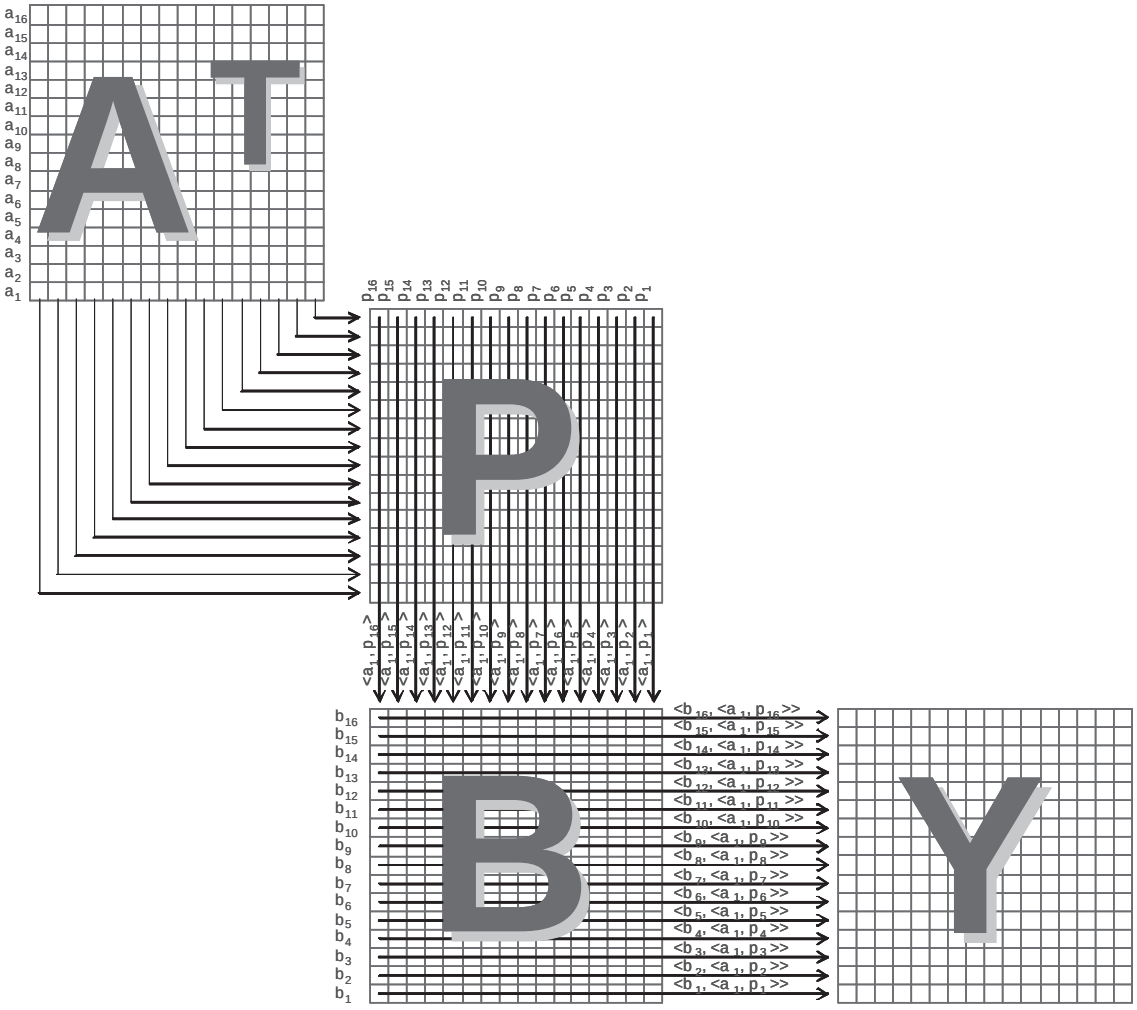


Figure 4.2. Matrix multiplication procedure on the separable transform imager.

4.2 Mathematically Equivalent Processing Model of Filtering Steps

The spatial filtering operations in the conventional prefiltering and derivative operation steps of a gradient-based method are not easily integrated because of a sequential and parallel structure in the prefiltering and derivative filtering steps. To achieve integrated prefiltering and derivative operation steps, we developed a new filtering scheme using convolution properties.

The cascaded filtering structure of the prefiltering and derivative operation can be parallelized by the convolution theorem. In Fig. 4.3, the filter outputs are summed for convenience in showing mathematical equivalence. The conventional structure can be expressed as

$$P = (D_x + D_y + D_t) * (S_t * (S_x * (S_y * I(x, y, t)))), \tag{4.6}$$

where S and D are smoothing and derivative filtering, respectively, and the subscripts indicates direction; and $I(x, y, t)$ represents a 3-D signal.

Based on convolution properties in [51], Eq. (4.6) can be expressed as

$$\begin{aligned} P = {} & D_y * (S_t * (S_x * (S_y * I(x, y, t)))) \\ & + D_x * (S_t * (S_x * (S_y * I(x, y, t)))) \\ & + D_t * (S_t * (S_x * (S_y * I(x, y, t)))) \end{aligned} \tag{4.7}$$

$$\begin{aligned} = {} & S_t * (S_x * (D_y * (S_y * I(x, y, t)))) \\ & + S_t * (D_x * (S_x * (S_y * I(x, y, t)))) \\ & + D_t * (S_t * (S_x * (S_y * I(x, y, t)))) \end{aligned} \tag{4.8}$$

$$\begin{aligned} = {} & S_t * (S_x * (SD_y * I(x, y, t))) \\ & + S_t * (SD_x * (S_y * I(x, y, t))) \\ & + SD_t * (S_x * (S_y * I(x, y, t))), \end{aligned} \tag{4.9}$$

where SD represents a combined filter for prefiltering and derivative filtering.

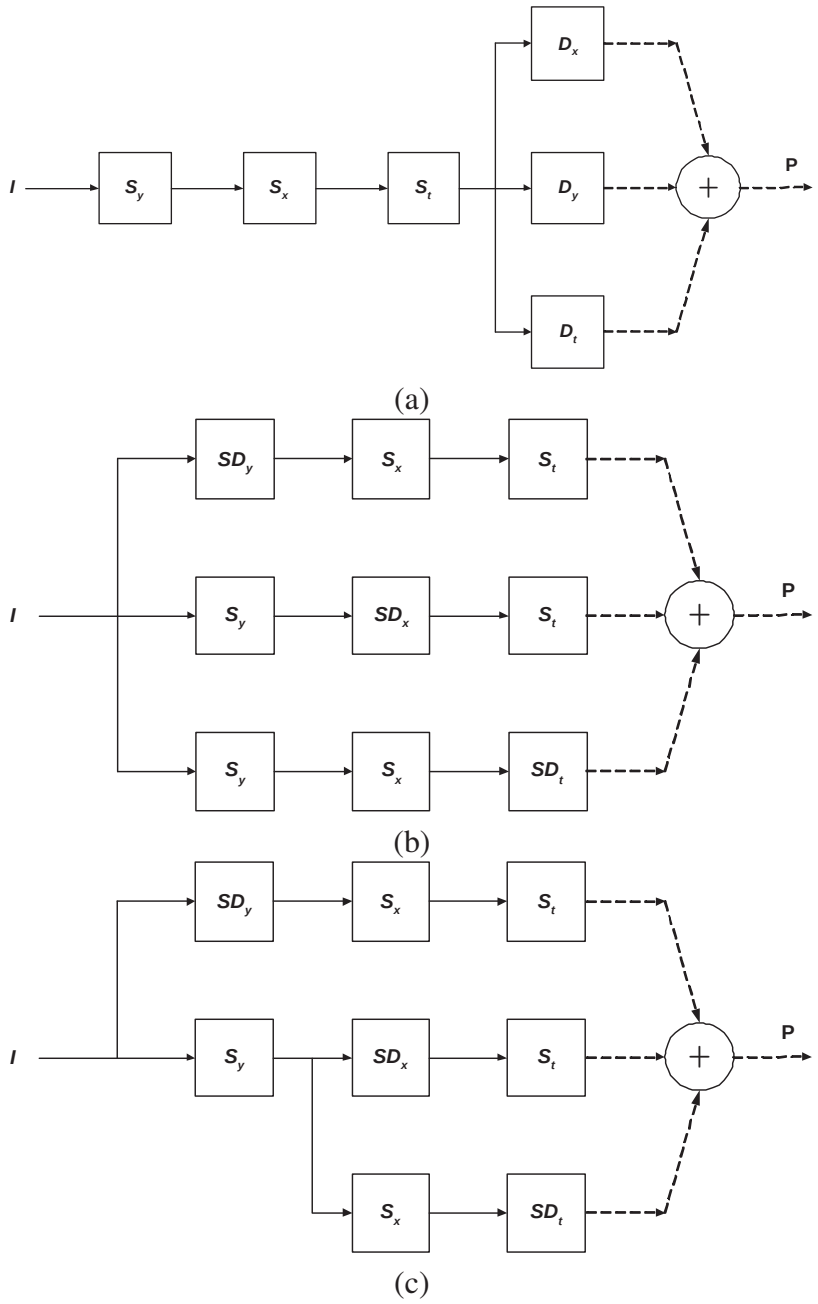


Figure 4.3. A set of equivalent expressions for prefiltering and derivative filtering. S is a prefiltering, D is a derivative filtering, and SD is a combined filter of prefiltering and derivative filtering.

Equation (4.9) is a fully parallelized version of Eq. (4.6). By parallelization, derivative values of x, y, and t directions can be calculated at the same time (see Fig. 4.3). This scheme reduces the number of steps of filtering from four to three.

Figure 4.3(b) is also same structure for Simoncelli's filter [14]. For Simoncelli's filter

case, SD should be replaced by D. Therefore, Fig. 4.3(c) can implement both conventional prefiltering and derivative filtering steps and Simoncelli's filter structure efficiently.

4.3 Checkerboard-Type Filtering

As illustrated in Fig. 4.3.(c) and Eq. (4.9), we can develop a fully parallelized system with a reduced number of filtering stages for conventional prefiltering and derivative filtering steps. Checkerboard-type filtering (CBTF) is a filtering algorithm to implement the structure of Fig. 4.3.(c) in a focal plane architecture [36]. The first step is to calculate prefiltering (S) and a combined filtering (SD) of prefiltering and derivative filtering (D) for the vertical direction in parallel. These values are assigned to different rows in an interlaced way. This means that only half of the total number of rows of image — odd rows in Fig. 4.4 — can be processed. The first step can be called the vertical step. The following step, which is applied to the result of the first step, is to perform S and SD for the horizontal direction in parallel. The result of the second step is assigned to different columns in an interlaced way. This means that only half of the total number of columns of the image, which is processed with the first step — odd columns in Fig. 4.4 — can be processed. We refer to the second step as a horizontal step. These two steps are for spatial transforms in image/video processing. Therefore, this spatial filtering algorithm can be applied for the separable transform imager. The final step is an independent filtering in parallel for each spatially transformed value with respect to the temporal direction. Figure 4.4 shows the block diagram of checkerboard-type filtering in an analog spatial transform imager for sub-sampled resolution OFE case, and Fig. 4.5 describes how derivative operations are performed in focal plane architecture if prefilterings are ignored. Because the system is a focal plane architecture having the same size as the image, the only way to represent the results of the three parallel processing for each pixel is through sub-sampling (see Fig. 4.4). This is helpful to maintain high/good fill-factor for imager. However, if we use more circuitry to support parallel processing per pixel in an analog spatial transform imager, full resolution

processing can be easily performed. CBTF can efficiently implement both Simoncelli's filter and conventional filtering structures on a focal plane imager. In addition, CBTF can move maximal amount of filtering to imager to exploit the functionality of the STI.

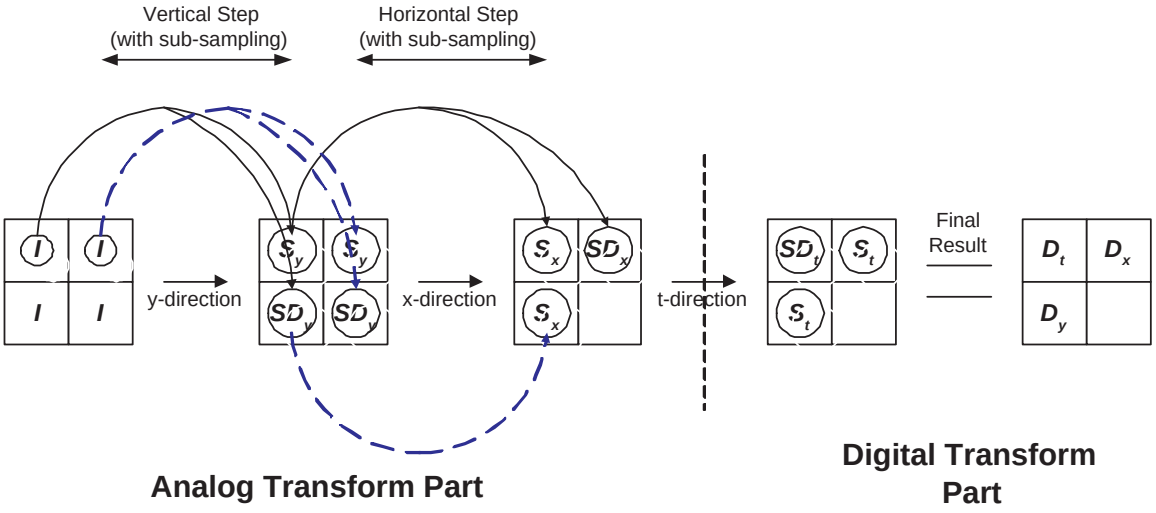


Figure 4.4. The block diagram of a checkerboard-type filtering procedure in a CADSP approach for sub-sampled resolution OFE case. I is a image signal, S is a prefiltering, D is a derivative filtering, and SD is a combined filter of prefiltering and derivative filtering. Subscripts indicate direction

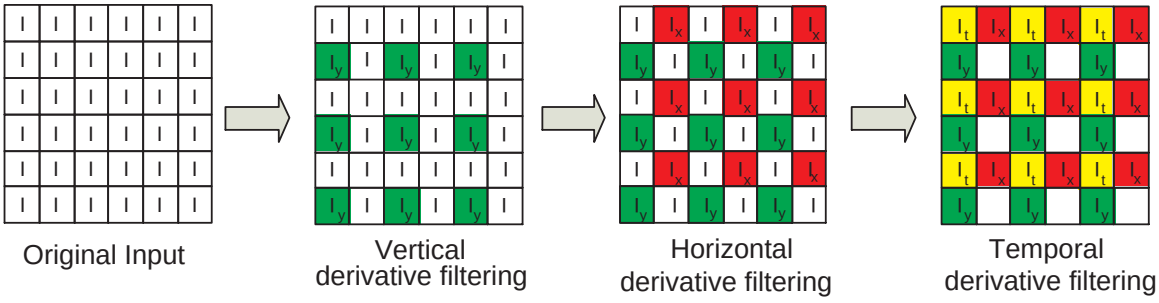


Figure 4.5. The block diagram of a checkerboard-type filtering in a whole focal plane architecture for sub-sampled resolution OFE case. I_x is a derivative with respect to x axis, I_y is a derivative with respect to y axis, and I_t is a derivative with respect to t axis. For convenience prefiltering is ignored to illustrate the checker-board type filtering in a whole imager

4.4 Comparison and Discussion

In this section, we compare the number of operations and memory access between the conventional structure (CS) and the fully parallel structure with data sharing (FPSDS) based on

a CADSP approach. For these comparisons, N and M are assumed as filter lengths for S and D, respectively. Since SD is the result of the convolution of S and D, the SD filter length is $N + M - 1$. Since most high performance filters of S and D for gradient-based optical flow algorithms have equal length for S and D, we assume N and M are equal [63], [43]. In each filtering block, we assume the required data for filtering is already in buffer/memory. In addition, each filtering block can generate an output by using serial/parallel input data supply and appropriate memory interleaving system. The image size is assumed to be $X \times Y$ and the on-chip/off-chip memory size is large enough to save the intermediate results after finishing the analog domain processing. Since the number of multiplications are almost same as that of additions and the power consumption and computational complexity of multiplication are higher than that of addition, we focus on the number of computation of multiplication for general FIR filtering operations. Then the total number of operations for CS is $(X \times Y) \times 3 \times (N + M)$. FPSDS has $(X \times Y) \times (2.5 \cdot N + M - 1)$ because FPSDS uses down-sampling.

Because we use our filtering algorithm on the separable transform imager, the CS and FPSDS will have the same expected power consumption and number of computations in our imager. Therefore, the controllable parts of computational complexity and memory access to reduce power consumption and computational complexity are digital domain. Since FPSDS needs one filtering step in the digital domain compared with that CS needs two filtering step in digital domain, FPSDS has more possibility to reduce power consumption and computational complexity in terms of the computational complexity and memory access.

In order to compare the CS and FPSDS, we assume the number of operations and power consumptions for 2-D spatial filtering in separable transform imager and ADC are same for CS and FPSDS to simplify our comparison. In fact, FPSDS needs only $3/4$ of the number of ADC operations of CS. Therefore, our assumption gives more penalty for FPSDS.

The total complexity in terms of number of computations and/or memory access can be

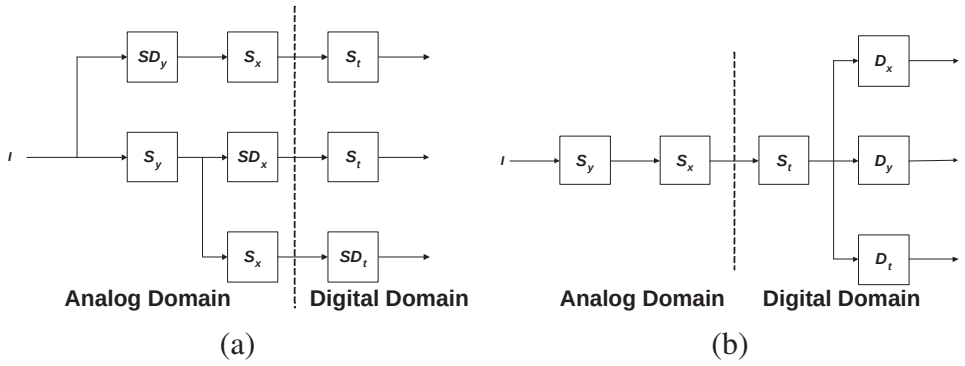


Figure 4.6. Analog and digital domain partition for conventional structure (CS) and fully parallel structure with data sharing (FPSDS). (a) Fully parallel structure with data sharing; (b) Conventional structure

declared as follows:

$$P_{Total(CS)} = P_{Analog(CS)} + P_{ADC(CS)} + P_{Digital(CS)}, \tag{4.10}$$

$$P_{Total(FPSDS)} = P_{Analog(FPSDS)} + P_{ADC(FPSDS)} + P_{Digital(FPSDS)}, \tag{4.11}$$

where $P_{Total(\cdot)}$, $P_{Analog(\cdot)}$, $P_{ADC(\cdot)}$, and $P_{Digital(\cdot)}$ indicates the total complexity of CS or $FPSDS$, the analog complexity of CS or $FPSDS$, the ADC complexity of CS or $FPSDS$, and the digital complexity of CS or $FPSDS$, respectively. From our assumptions, $P_{Analog(CS)} = P_{Analog(FPSDS)}$ and $P_{ADC(CS)} = P_{ADC(FPSDS)}$. Therefore, our concern for efficient filtering strategy is dependent on $P_{Digital(CS)}$ and $P_{Digital(FPSDS)}$, which are dependent on the number of operations and memory accesses.

4.4.1 FPSDS vs. CS without Post-Processing after OFE

We consider two cases for the comparison between $FPSDS$ and CS. If $FPSDS$ and CS are compared directly, $FPSDS$ has advantages for the number of computations and memory accesses. One reason comes from the fact that only $3/4$ of the total number of samples per frame are processed after finishing the spatial filtering step, and another reason is due to the reduced filtering stage from two steps in CS into one step in $FPSDS$. $P_{Digital(CS)}$ and $P_{Digital(FPSDS)}$ are dependent on the number of computations and memory access. The total

number of computations per frame based on $CBTF$ and CS are

$$C_{CS} = X \cdot Y(N + 3M) \tag{4.12}$$

$$C_{FPSDS} = X \cdot Y\left(\frac{1}{2} \cdot N + \frac{1}{4} \cdot (N + M - 1)\right) \tag{4.13}$$

In addition, the number of memory access, which includes memory read and write operations per frame, are

$$M_{CS} = Mem_{wr(CS)} + Mem_{rd(CS)} = X \cdot Y\Big(5 + (N + 3M)\Big) \tag{4.14}$$

$$M_{FPSDS} = Mem_{wr(FPSDS)} + Mem_{rd(FPSDS)} = X \cdot Y\left(\frac{6}{4} + \frac{1}{4} \cdot (3N + M - 1)\right) \tag{4.15}$$

For any N and M, C_{FPSDS} is better than C_{CS}. And M_{FPSDS} is also efficient than M_{CS}. Therefore, we can conclude that $CBTF$ is faster algorithm than general prefiltering and derivative filtering scheme for the focal-plane STI. The downside of $FPSDS$ is that it results in optical flow estimates with lower density.

Even though the full optical flow density is possible using CS scheme, the result of spatial filtering of CS scheme after the STI can be subsampled horizontally and vertically ($CS_{quarter}$). This is another case for comparison between CS and $FPSDS$. $CS_{quarter}$ makes the number of pixel points for optical flow estimation to the same as the proposed $CBTF$ algorithm. In this case, the total number of calculations are

$$C_{CS_{quarter}} = X \cdot Y\left(\frac{1}{4} \cdot N + \frac{3}{4} \cdot M\right) \tag{4.16}$$

$$C_{FPSDS} = X \cdot Y\left(\frac{1}{2} \cdot N + \frac{1}{4} \cdot (N + M - 1)\right) \tag{4.17}$$

In addition, the number of memory access, which includes memory read and write operations, are

$$M_{CS_{quarter}} = Mem_{wr(CS_{quarter})} + Mem_{rd(CS_{quarter})} = X \cdot Y\Big(\frac{5}{4} + \frac{1}{4} \cdot (N + 3M)\Big) \tag{4.18}$$

$$M_{FPSDS} = Mem_{wr(FPSDS)} + Mem_{rd(FPSDS)} = X \cdot Y\left(\frac{6}{4} + \frac{1}{4} \cdot (3N + M - 1)\right) \tag{4.19}$$

For $N = M$ case, the computational complexity and number of memory access are almost same for $CS_{quarter}$ and $FPSDS$ although C_{FPSDS} is slightly better than $CS_{quarter}$. Based on above observations, there is a slight computational complexity advantage for $FPSDS$ than that of $CS_{quarter}$ with approximately same number of memory access for both cases. The latency is the main difference between two schemes. As we observed before, the number of filtering steps of $FPSDS$ is less than $CS/CS_{quarter}$. If we prefer to design fast execution or low latency, $FPSDS$ is better choice than $CS_{quarter}$. In addition, no CS can be used to implement Simoncelli's filter efficiently.

Table 4.1. The comparison between a conventional structure (CS) and a fully parallel structure with data sharing (FPSDS) regarding the number of computations, memory accesses, and filtering steps in digital domain per image pixel. The number of filtering step in analog domain is two for all cases.

Filtering structure	No. of computations per image pixel	No. of Mem accesses per image pixel	No. of filtering steps in digital domain
CS	$(N + 3M)$	$(5 + (N + 3M))$	2
$FPSDS$	$\left(\frac{1}{2} \cdot N + \frac{1}{4} \cdot (N + M - 1)\right)$	$\left(\frac{6}{4} + \frac{1}{4} \cdot (3N + M - 1)\right)$	1
$CS_{quarter}$	$\left(\frac{1}{4} \cdot N + \frac{3}{4} \cdot M\right)$	$\left(\frac{5}{4} + \frac{1}{4} \cdot (N + 3M)\right)$	2

Table 4.1 shows that $FPSDS$ has low or similar complexity with the reduced filtering steps compared to CS and $CS_{quarter}$. The next interesting facts of $FPSDS$ and CS are their performance. We performed some simulations based on $FPSDS$ and CS since the performance between $FPSDS$ and $CS_{quarter}$ are same. For simulation, we use Gaussian filtering with $\sigma = 1.5$ as pre-filtering (S) step. For derivative filter, Barron's 5-tap filter and series-based 11-tap filters are used as follows:

$$D_{barron} = \frac{1}{12}[-1, 8, 0, -8, 1] \tag{4.20}$$

$$D_{series} = \frac{1}{2520}[-2, 25, -150, 600, -2100, 0, 2100, -600, 150, -25, 2]. \tag{4.21}$$

For Gaussian prefiltering with $\sigma = 1.5$, Barron's 5-tap filter has $N > M$ and the series-based 11-tap filter has $N = M$.

Test image sequences are translational and diverging tree image sequences with size 150×150 in Fig. 5.4. The performance of the algorithm is tested based on the angular error between ground truth and the estimated motion fields.

Table 4.2. Angular error performance of LS-OFE based on CS and FPSDS [5]. Avg and Std represent average and standard deviation, respectively. And density means the optical flow density.

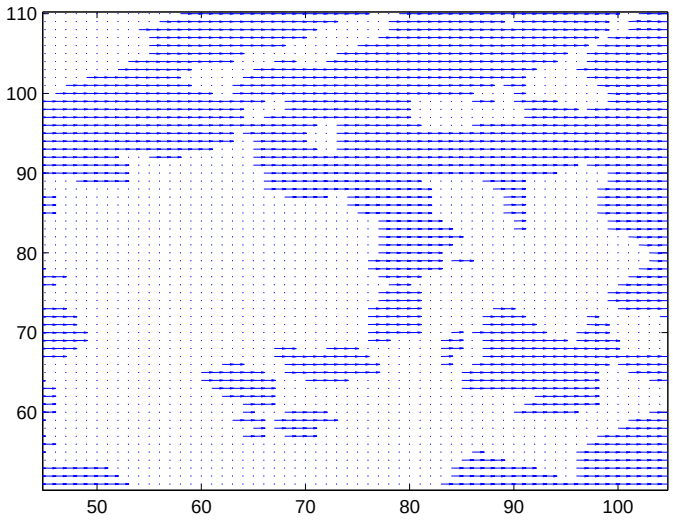
	Translating Tree			Diverging Tree		
Filtering Scheme	Avg Error	Std	Density	Avg Error	Std	Density
FPSDS (Barron)	0.58°	0.58°	15%	2.10°	1.96°	16%
CS (Barron)	0.64°	0.66°	46%	2.02°	2.06°	57%
FPSDS (Series)	0.46°	0.45°	16%	2.16°	2.02°	16%
CS (Series)	0.52°	0.55°	50%	1.99°	1.99°	54%

Table 4.2 shows the performance with thresholding in terms of angular error . As we can observe, the D_{series} filter has better performance than the D_{barron} for CS. For $FPSDS$ case, the situation is slightly different. This can be caused by subsampling. Overall there are only slight differences for angle error performance between CS and $FPSDS$. However, the optical flow density of $FPSDS$ is between one-third and one-quarter of that of CS (See Fig. 4.7 and Fig. 4.8). Therefore, $FPSDS$ is advantageous to use for applications without the requirement of high optical flow density because $FPSDS$ has lower computational complexity and memory accesses than CS with similar angle error performance.

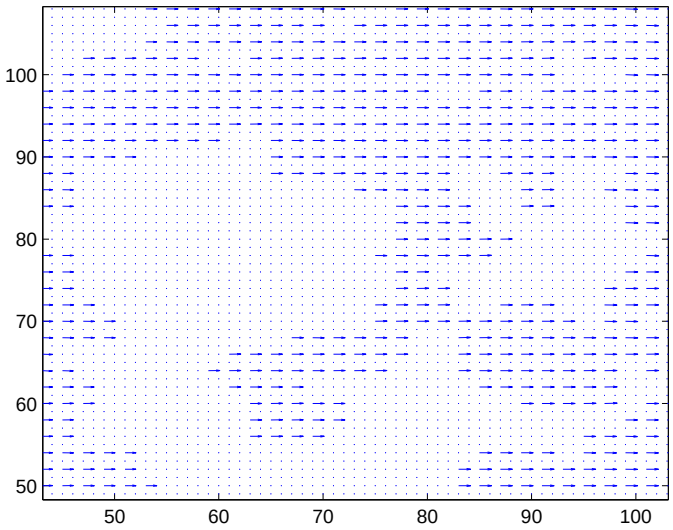
4.5 Conclusions

In this chapter, we proposed a checkerboard-type filtering algorithm for a efficient gradient-based optical flow estimation (OFE) system. Checkerboard-type filtering (CBTF) is a filtering algorithm to implement fully parallel structure with data sharing (FPSDS) in a focal

plane architecture. This makes FPSDS is efficient in terms of computational complexity and the number of memory accesses compared to conventional filtering structure in STI-based system design. CBTF can move a significant amount of the filtering to the imager to exploit the functionality of the STI. In addition, CBTF can implement both Simoncelli's filter and conventional prefitering and derivative filter steps efficiently on a focal plane imager. There are only slight performance differences for angular error performance between FPSDS and CS in a LS-based OFE algorithm. However, the optical flow density of FPSDS is between one-third and one-quarter of that of CS. This makes FPSDS profitable for applications without the requirement of high optical flow density.

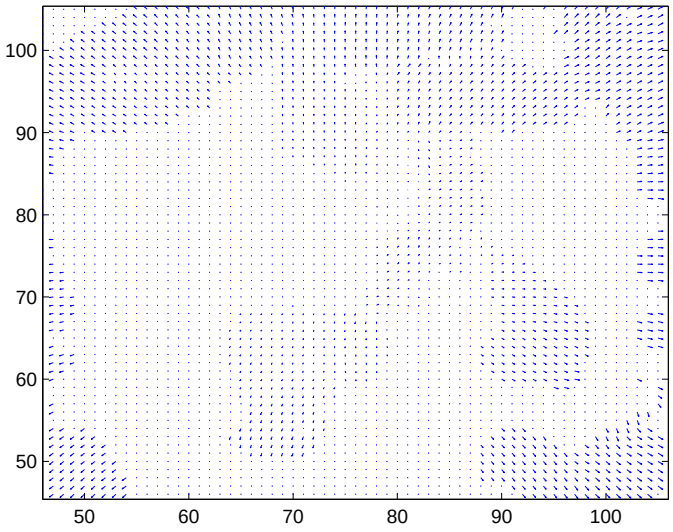


(a)

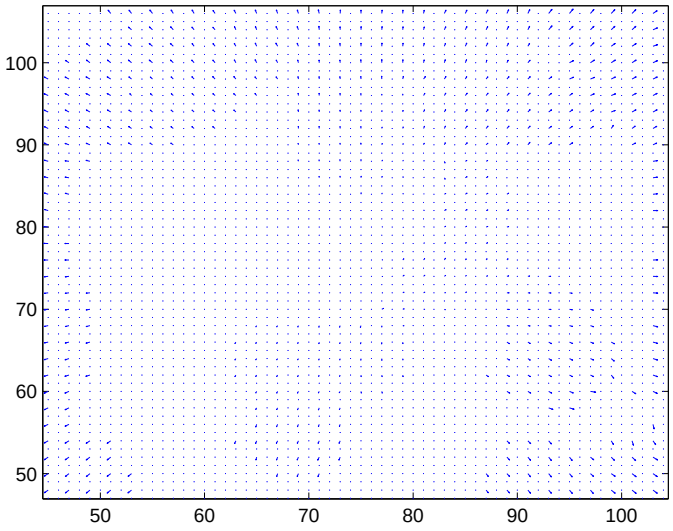


(b)

Figure 4.7. Optical flow field results by LS-OFE using $\sigma = 1.5$ for translational tree image sequence. (a) Optical flow field using CS; (b) Optical flow field using FPSDS



(a)



(b)

Figure 4.8. Optical flow field results by LS-OFE using $\sigma = 1.5$ for diverging tree image sequence. (a) Optical flow field using CS; (b) Optical flow field using FPSDS

CHAPTER 5

LOW-COMPLEXITY LS-OFE ALGORITHMS FRAMEWORKS USING RECURSION

In this chapter, we propose low-complexity LS-OFE algorithm frameworks using recursion and some performance and real-time experiments are compared with conventional LS-OFE. Low-complexity LS-OFE algorithms are attempted to reduce the redundancy of LS computations for OFE between adjacent pixels. This chapter is composed of five sections. Basic redundancy analysis of LS-OFE is described in section 5.1. Parametric motion modeling for LS-OFE is explained in section 5.2. Low-complexity LS-OFE algorithm frameworks using recursion is presented in section 5.3. Experimental comparison with respect to performance and computational complexity are discussed in section 5.4. Finally, conclusion is described in section 5.5.

5.1 Basic Redundancy Analysis of LS-OFE

Based on simulation results in [5] and [44], a gradient-based optical flow estimation (OFE) algorithm using a weighted least-squares (LS) technique is one of most successful solutions in terms of performance and the number of operations. However, it has some redundancies for calculating successive LS among adjacent pixels. As video frame rate and/or video format size per each frame increase, it is important to find more efficient ways of estimating motion information.

Previous work regarding efficient LS-related optical flow estimation algorithms are reported in [43], [55]. and [18]. Liu, *et al.* improved the performance using an adaptive structure tensor and an affine parametric motion model. However, considering the computational complexity for embedded real-time system design, a gradient-based OFE using a

LS technique can be a better choice because of low-complexity with comparable performance. Rav-Acha and Peleg proposed an efficient way to implement Lukas-Kanade algorithm, which is an iterative LS-type algorithm using warping. This efficient algorithm does not help for non-iterative LS solutions. Fleet and Langley employed recursive temporal filtering using infinite impulse reaponse (IIR) filtering for a LS technique of local first-order constraints to reduce the number of temporal frames and filtering operations [18]. Even though Fleet and Langley achieved an efficient algorithm using temporal recursive filtering with comparable performance [5], a potential problem is caused from IIR filtering. The IIR approach is more sensitive to finite precision arithmetic, as is commonly required for area/power efficient implementations.

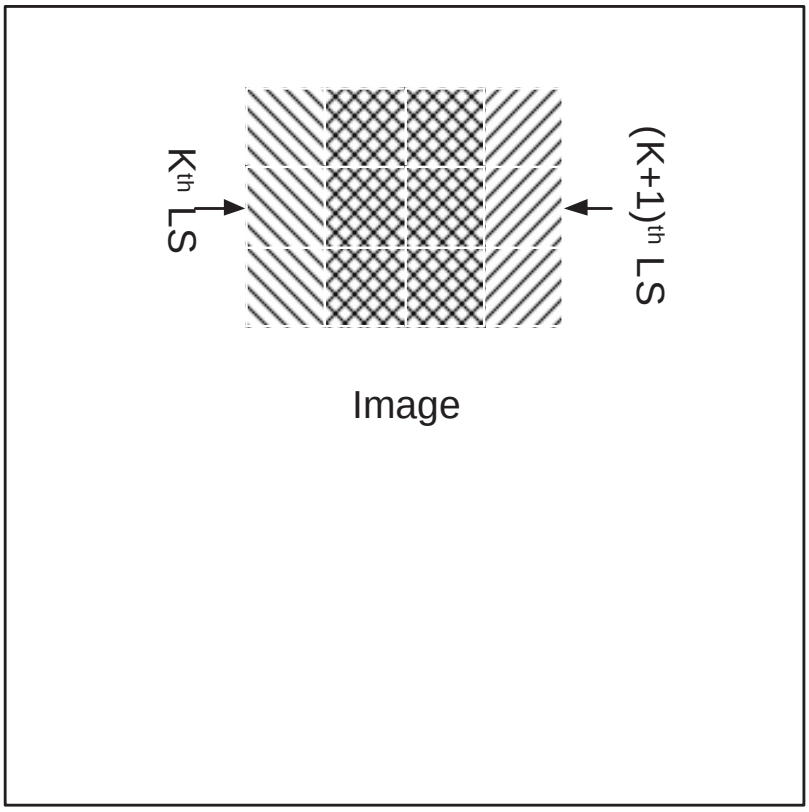


Figure 5.1. Overlapped data between LS windows for block-wise RLS.

Mathematically conventional LS-OFE can be described as follows:

$$\arg\min_{\vec{v}} \sum_{\vec{x}\in\Omega} W(\vec{x},t)[\vec{I_s}\cdot\vec{v}+I_t(\vec{x},t)]^2, \tag{5.1}$$

where $W(\vec{x},t)$ is a weighting window in the neighborhood Ω, $\vec{I_s}=(I_x,I_y)^T$ and I_x, I_y, and I_t are spatial and temporal derivatives respectively of an image I, and $\vec{v}=(v_x,v_y)^T$ is a motion vector for a dense motion field, optical flow. In Eq. 5.1, $\vec{I_s}\cdot\vec{v}+I_t(\vec{x},t)$ denotes the gradient-based OFE constraint equation. The basic idea of original LS-OFE is to compose the locally over-determined system for finding the unique solution of under-determined OFE constraint equations. The motion model used for typical LS-OFE is the locally constant model within the window. To consider the correlation of motion vectors among adjacent pixels, weighted window is employed. This transforms the problem of estimating motion vector into a LS problem. LS-OFE achieves the high performance with low-complexity compared with other OFE algorithms. Even though LS-OFE is one of successful low-complexity algorithms, there are some redundancy in LS computations for OFE among adjacent pixels. This can be easily observed in Fig. 5.1, which shows the adjacent two LS blocks with 3×3 window for over-determine systems. Figure 5.1 shows that only one column values of spatial and temporal derivatives within the window are different between two LS blocks except weighting. This indicates that we can improve the computational complexity efficiency by re-using the derivative data in the overlapped area.

5.2 Parametric Motion Modeling for LS-OFE

Typical LS-OFE used the constant motion assumption within local neighborhood. This can be regarded as a locally constant motion model. The motion model is acceptable for translational motion in image sequence. However, this model has limitation to capture high order movements within the local neighborhood such as rotation, stretch, shear, and divergence. These high order motion can be modeled efficiently by parametric model to reduce the degree of freedom of model function.

In optical flow estimation literature, there are several parametric motion models. Popular four models can be summarized as follows:

1 Constant Model: Velocity is assumed as constant in local image neighborhood, which is used for conventional LS-OFE [5].

$$u(x, y) = p_1 \tag{5.2}$$

$$v(x, y) = p_2 \tag{5.3}$$

2 First Order Model (Affine Model): First order variation is assumed within the local neighborhood. This is the most popular model [8].

$$u(x, y) = p_1 x + p_2 y + p_3 \tag{5.4}$$

$$v(x, y) = p_4 x + p_5 y + p_6 \tag{5.5}$$

3 Second Order Model: First and second order variations are assumed within local neighborhood [74].

$$u(x, y) = p_1 x^2 + p_2 xy + p_3 y^2 + p_4 x + p_5 y + p_6 \tag{5.6}$$

$$v(x, y) = p_7 x^2 + p_8 xy + p_9 y^2 + p_{10} x + p_{11} y + p_{12} \tag{5.7}$$

4 Planar Model: A special case of second order model with the assumption that local surface is planar [74].

$$u(x,y) = p_1x^2 + p_2xy + p_3x + p_4y + p_5 \tag{5.8}$$

$$v(x,y) = p_1xy + p_2y^2 + p_6x + p_7y + p_8 \tag{5.9}$$

Even though typical LS-OFE is based on block-wise constant motion model, a variety of motion modelings can be incorporated into LS-OFE. In order to modify the conventional LS-OFE with other motion model, some parts of equations of the LS-OFE are needed to be parameterized. The main part to change is gradient-based OFE constraint equation, $\vec{I_s} \cdot \vec{v} + I_t(\vec{x}, t)$, in Eq. 5.1. To incorporate other motion models in the constraint equation, $\vec{I_s}$ and $\vec{v}$ should be redefined as follows:

$$\vec{I_s}(\vec{x}, t) = D_{1\times2}, \quad \vec{v} = M_{2\times2} \cdot \vec{p}, \tag{5.10}$$

where

$$D_{1\times2} = [I_x, I_y],\ M_{2\times2} = \begin{bmatrix} 1 & 0 \\ & \\ 0 & 1 \end{bmatrix},\ \vec{p} = [p_1, p_2]^T. \tag{5.11}$$

Therefore, the OFE constraint equation can be expressed as

$$\vec{I_s} \cdot \vec{v} = -I_t(\vec{x}, t) \tag{5.12}$$

$$D_{1\times2}M_{2\times2} \cdot \vec{p} = -I_t(\vec{x}, t). \tag{5.13}$$

By changing M and $\vec{p}$ depending on the parametric motion model, we can implement LS-OFE algorithms with different motion models (See Table 5.1). Equations (5.10) and (5.11) are for the LS-OFE algorithm with the constant model model. We use the notation of Eq. (5.10) for other parametric cases. For convenience, the motion model matrix equations are only shown using block-wise sub-matrices. This is assumed over this chapter implicitly.

Table 5.1. The parametric motion modelings for OFE algorithms.

1. Constant motion modeling:
   $\vec{I}_s(\vec{x}, t) = D, \vec{v} = M\vec{p}$
   $D = \begin{bmatrix} I_x & I_y \end{bmatrix}, \quad M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \vec{p} = [p_1, p_2]^T$
   $DM\vec{p} = \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$

2. Affine motion modeling:
   $\vec{I}_s(\vec{x}, t) = D, \vec{v} = M\vec{p}$
   $D = \begin{bmatrix} I_x & I_y \end{bmatrix}, \quad M = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix},$
   $\vec{p} = [p_1, p_2, p_3, p_4, p_5, p_6]^T$
   $DM\vec{p} = \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_6 \end{bmatrix}$

3. Second order motion modeling:
   $\vec{I}_s(\vec{x}, t) = D, \vec{v} = M\vec{p}$
   $D = \begin{bmatrix} I_x & I_y \end{bmatrix}, \quad M = \begin{bmatrix} x^2 & xy & y^2 & 0 & 0 & 0 & x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x^2 & xy & y^2 & 0 & 0 & 0 & x & y & 1 \end{bmatrix},$
   $\vec{p} = [p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}]^T$
   $DM\vec{p} = \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} x^2 & xy & y^2 & 0 & 0 & 0 & x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x^2 & xy & y^2 & 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_{12} \end{bmatrix}$

4. Planar motion modeling:
   $\vec{I}_s(\vec{x}, t) = D, \vec{v} = M\vec{p}$
   $D = \begin{bmatrix} I_x & I_y \end{bmatrix}, \quad M = \begin{bmatrix} x^2 & xy & x & y & 1 & 0 & 0 & 0 \\ xy & y^2 & 0 & 0 & 0 & x & y & 1 \end{bmatrix},$
   $\vec{p} = [p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8]^T$
   $DM\vec{p} = \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} x^2 & xy & x & y & 1 & 0 & 0 & 0 \\ xy & y^2 & 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_8 \end{bmatrix}$

5.3 Low-Complexity LS-OFE Algorithm Frameworks using Recursion

Even though the gradient-based optical flow estimation (OFE) algorithm using a least-squares (LS) technique is one of most successful solutions in terms of performance and computational complexity, it still has some redundancies for calculating successive LS among adjacent pixels. Therefore, there are some rooms to improve the computational

complexity of LS-OFE. To achieve computational complexity reduction, we introduce two different approaches by reuse of matrix elements. The first one is a LS-OFE framework using spatially recursive matrix refinement. This framework can reuse the overlapped spatial and temporal derivative values to reduce the computational complexity. Another one is a spatially recursive optical flow estimation framework using adaptive filtering. This framework exploits properties of matrix inversion and adaptive filtering to reduce the computational complexity of matrix inversion.

5.3.1 A LS-OFE Framework using Recursive Matrix Refinement (RMR-OFE)

Typical LS-OFE can be described as follows:

$$\arg\min_{\vec{v}} \sum_{\vec{x}\in\Omega} W(\vec{x},t)[\vec{I_s}\cdot\vec{v}+I_t(\vec{x},t)]^2, \tag{5.14}$$

where $W(\vec{x},t)$ is a weighting window in the neighborhood Ω, $\vec{I_s}=(I_x,I_y)$ and I_x, I_y, and I_t are spatial and temporal derivatives respectively of an image I, and $\vec{v}=(v_x,v_y)^T$ is a motion vector for a dense motion field, optical flow.

If $W=I$, Eq. (5.14) can be represented as following form:

$$A^TA\vec{v}=A^T\vec{b}, \tag{5.15}$$

$$[\vec{I_s}^T(\vec{x_1},t),\cdots\vec{I_s}^T(\vec{x_n},t)]\begin{bmatrix}\vec{I_s}(\vec{x_1},t)\\ \vdots \\ \vec{I_s}(\vec{x_n},t)\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=-[\vec{I_s}^T(\vec{x_1},t),\cdots\vec{I_s}^T(\vec{x_n},t)]\begin{bmatrix}\vec{I_t}(\vec{x_1},t)\\ \vdots \\ \vec{I_t}(\vec{x_n},t)\end{bmatrix}, \tag{5.16}$$

$$\begin{bmatrix}\sum I_x^2 & \sum I_xI_y\\ \sum I_xI_y & \sum I_y^2\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=\begin{bmatrix}\sum I_xI_t\\ \sum I_yI_t\end{bmatrix}, \tag{5.17}$$

where

$$A^T=[\vec{I_s}(\vec{x_1},t),\cdots,\vec{I_s}(\vec{x_n},t)] \tag{5.18}$$

$$\vec{b}=-[\vec{I_t}(\vec{x_1},t),\cdots,\vec{I_t}(\vec{x_n},t)]^T \tag{5.19}$$

Equation (5.17) is the matrix form of LS-OFE using locally constant motion model. This matrix form can be expressed using $A\vec{v} = DM\vec{p}$ from Eq. (5.13) as follows:

$$A^T A\vec{v} = A^T\vec{b}, \tag{5.20}$$

$$(DM)^T(DM)\vec{p} = (DM)^T\vec{b}, \tag{5.21}$$

$$M^T D^T DM\vec{p} = M^T D^T\vec{b}. \tag{5.22}$$

Equation (5.22) is the general LS-OFE matrix form with parametric motion modeling. Because most data of successive LS operations between adjacent pixels are overlapped except the amount of one column size data bounded by the neighborhood window (See Fig. 5.2), we can reuse data in A or DM matrices. To exploit the data overlap property in successive LS computations for OFE, we can make a LS-OFE matrix refinement scheme using data reuse in successive pixel locations. The matrix refinement process is composed of two terms: Incoming data set, which is a new column data set only for current neighborhood window, and outgoing data set, which is a old column data set to be erased from the current LS calculations. In matrix form, the matrix refinement process can be expressed as follows:

$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}, \tag{5.23}$$

$$(A^T A)_{+\Delta} = A^T A + \begin{bmatrix} \sum I_{x(+\Delta)}^2 & \sum I_{x(+\Delta)} I_{y(+\Delta)} \\ \sum I_{x(+\Delta)} I_{y(+\Delta)} & \sum I_{y(+\Delta)}^2 \end{bmatrix}, \tag{5.24}$$

$$(A^T A)_{-\Delta} = (A^T A)_{+\Delta} - \begin{bmatrix} \sum I_{x(-\Delta)}^2 & \sum I_{x(-\Delta)} I_{y(-\Delta)} \\ \sum I_{x(-\Delta)} I_{y(-\Delta)} & \sum I_{y(-\Delta)}^2 \end{bmatrix}, \tag{5.25}$$

where $A^T A$ is the matrix for LS-OFE, $+\Delta$ is the portion of incoming new data set, and $-\Delta$ is the portion of outgoing old data set.

By using the matrix refinement scheme, we can minimize the number of multiplications and additions for making linear system matrices. After each matrix refinement process, the

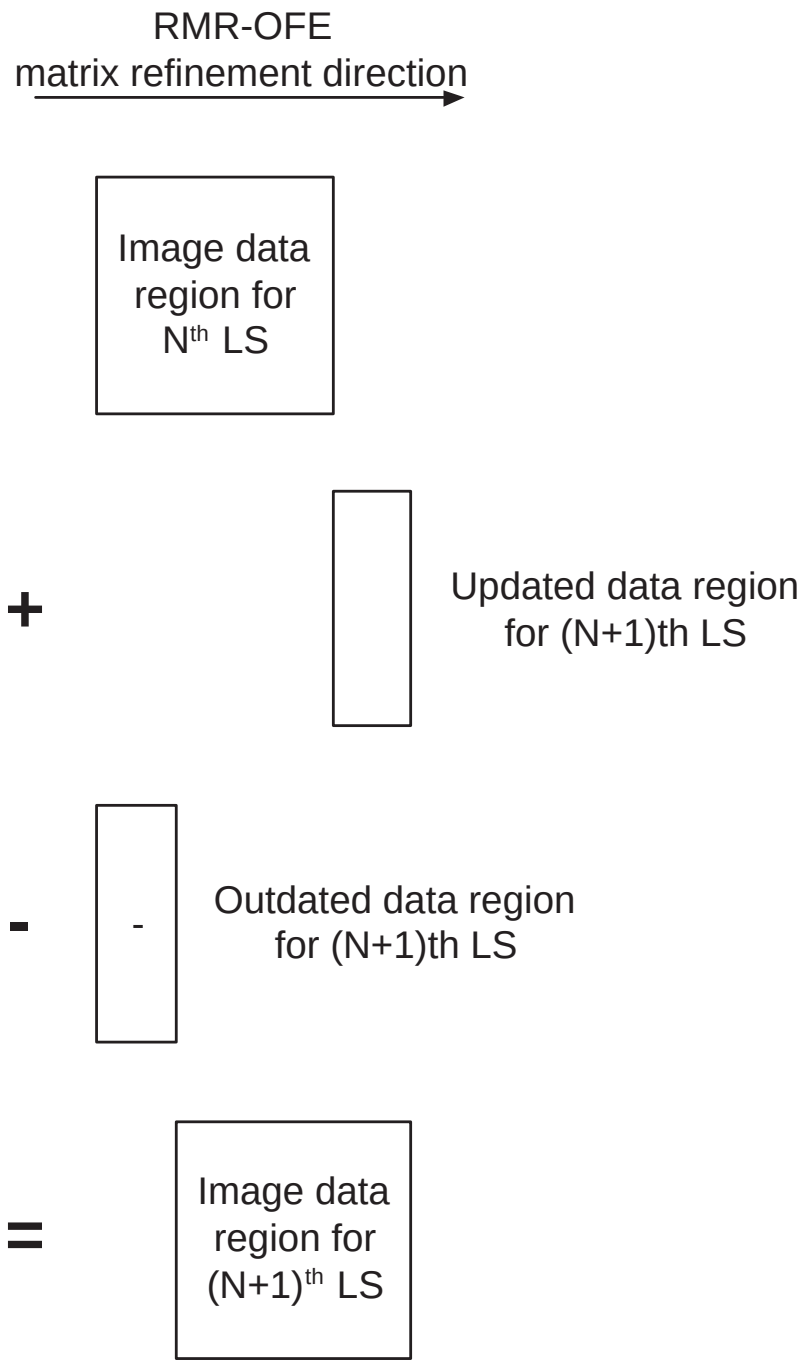


Figure 5.2. Block diagram for the matrix refinement procedure of RMR-OFE.

general LS computation is required to solve linear system equations. This scheme can be easily applicable to other motion modeling cases by using proper modeling matrix in Table 5.1.

5.3.2 A Spatially Recursive LS Optical Flow Estimation Framework using Adaptive Filtering (SR-OFE)

As another framework for reducing the computational complexity of LS-OFE, we can model LS-OFE as an adaptive filtering problem using a sliding window RLS technique. First we model the least-squares error function.

$$\varepsilon = \sum_{\vec{x}\in\Omega} |e(\vec{x}, t)|^2, \tag{5.26}$$

where Ω is a region of interest and

$$e(\vec{x}, t) = d(\vec{x}, t) - y(\vec{x}, t) = d(\vec{x}, t) - \vec{f}^T \cdot \vec{w}. \tag{5.27}$$

In Eq. (5.27), $d(\vec{x}, t)$ is the desired signal, $\vec{f}$ is the input signal for the FIR adaptive filter, and $\vec{w}$ is the filtering coefficient of the FIR adaptive filter. This set-up can transform the LS problem into adaptive filtering framework.

Based on Eq. (5.14), (5.10), and (5.27), we can model ε as follows:

$$\begin{aligned} \varepsilon &= \sum_{\vec{x}\in\Omega} W(\vec{x}, t)[\vec{I_s} \cdot \vec{v} + I_t(\vec{x}, t)]^2 = \sum_{\vec{x}\in\Omega} |e(\vec{x}, t)|^2 \\ &= \sum_{\vec{x}\in\Omega} |d(\vec{x}, t) - \vec{w}^T \cdot \vec{f}|^2, \end{aligned} \tag{5.28}$$

where $d(\vec{x}, t) = -I_t(\vec{x}, t)$, $\vec{f}^T(i) = \vec{I_s}$, and $\vec{w} = \vec{p}$. We use $W = I$ to match the adaptive filtering framework with a LS solution in Eq. (5.28).

Equation (5.28) can be represented in a matrix form as follows:

$$\mathbf{R}_f(n)\vec{w} = \vec{r}_{df}(n), \tag{5.29}$$

where $\mathbf{R}_f(n) = A^T A$, $\vec{w} = \vec{v}$, and $\vec{r}_{df}(n) = A^T\vec{b}$ in Eq. 5.15 and Eq. 5.29.

The gradient-based optical flow estimation (OFE) algorithm using a least-squares (LS) technique can be mapped into a sliding window RLS algorithm with Eq (5.28). We simplify the notation of a spatial and temporal axis format into an 1-D spatial axis form for illustration in Fig. 5.3 and Eq (5.30). Sliding window RLS is composed of two steps [27].

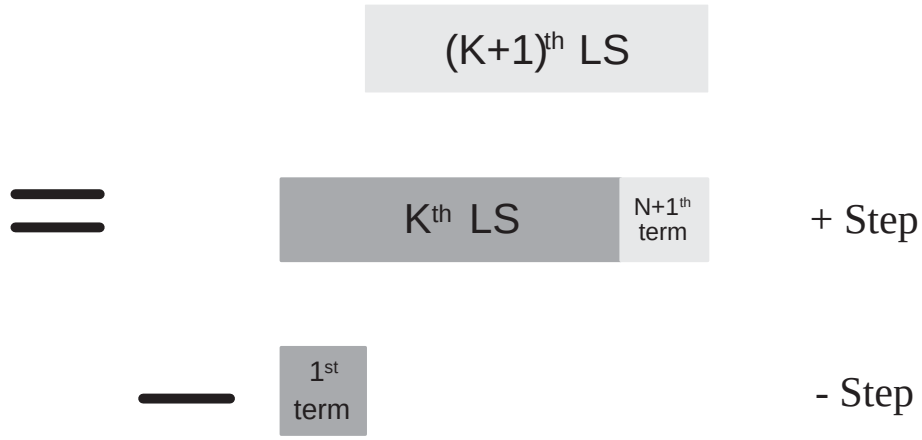


Figure 5.3. Two-step sliding window algorithm description.

The first step is a growing window RLS. This can be called the '+' step in Fig. 5.3 because this step adds a new data point for least-squares (LS). To perform LS for the same number of data points, we need to remove the effect of the oldest data point. We can call this step as the '-' step in Fig. 5.3 because this step reduces the window. Mathematically the sliding window algorithm can be regarded as two-step LS based on the previous local-window LS result. If we assume that the LS of previous pixel position for OFE is already calculated, the new LS can be calculated as follows (See Figure 5.3):

$$\min_{\vec{w}(K)} \sum_{1}^{N} |e(i)|^2 \xrightarrow{'+'\,step} \min_{\vec{w^+}(K+1)} \sum_{1}^{N+1} |e(i)|^2 \xrightarrow{'-'\,step} \min_{\vec{w}(K+1)} \sum_{2}^{N+1} |e(i)|^2, \tag{5.30}$$

where the '+' step is a growing window RLS step and the '-' step is a reducing window RLS step. The sliding window algorithm can turn $O(n^3)$ of LS into $O(n^2)$, where n is the 1-D size of a square matrix.

By using the sliding window RLS algorithm and the relationships in Eq (5.28), we can make a spatially recursive optical flow estimation framework. The algorithm is described in Table 5.2.

Table 5.2. A spatially recursive optical flow estimation framework using adaptive filtering and sliding window RLS techniques. We simplify the notation of the spatial and temporal axis format into a 1-D spatial axis form for illustration.

Fast LS-OFE framework using the two-step sliding window RLS

1. Signal modeling:

$d(i) = -I_t(\vec{x}, t)$, $\vec{f}^T(i) = \vec{I}_s$, and $\vec{w} = \vec{p}$

$e(i) = d(i) - y(i) = d(i) - \vec{f}^T(i) \cdot \vec{w}$

$e(n) = \sum_{\vec{x} \in \Omega} W(\vec{x}, t)[\vec{I}_s \cdot \vec{v} + I_t(\vec{x}, t)]^2 = \sum_{i=0}^{n} \lambda^{n-i} |e(i)|^2 = \sum_{i=0}^{n} \lambda^{n-i} |d(i) - \vec{w}^T \cdot \vec{x}|^2$

$\mathbf{R}_f(n) = \sum \vec{f}(i) \cdot \vec{f}^T(i), \quad \vec{r}_{df}(n) = \sum d(i)\vec{f}(i)$, $\mathbf{P}(n) = \mathbf{R}_f^{-1}(n)$

2. Algorithm

For $n = 0...N$

Do growing window RLS {

a) $\vec{z}(n) = \mathbf{P}(n-1)\vec{f}(n)$

b) $\vec{g}(n) = \frac{1}{1+\vec{f}^T(n)\cdot\vec{z}(n)}\vec{z}(n)$

c) $\alpha(n) = d(n) - \vec{w}^T(n-1) \cdot \vec{f}(n)$

d) $\vec{w}^+(n) = \vec{w}(n-1) + \alpha(n)\vec{g}(n)$

e) $\mathbf{P}^+(n) = \mathbf{P}(n-1) - \vec{g}(n) \cdot \vec{z}^T(n)$

}

Do reducing window RLS {

f) $\vec{z}^+(n) = \mathbf{P}^+(n)\vec{f}(n-L-1)$

g) $\vec{g}_+(n) = \frac{1}{1-\vec{f}^T(n-L-1)\cdot\vec{z}^+(n)}\vec{z}^+(n)$

h) $\alpha^+(n) = d(n-L-1) - \vec{f}^T(n-L-1) \cdot \vec{w}^+(n)$

i) $\vec{w}(n) = \vec{w}^+(n) - \alpha^+(n)\vec{g}^+(n)$

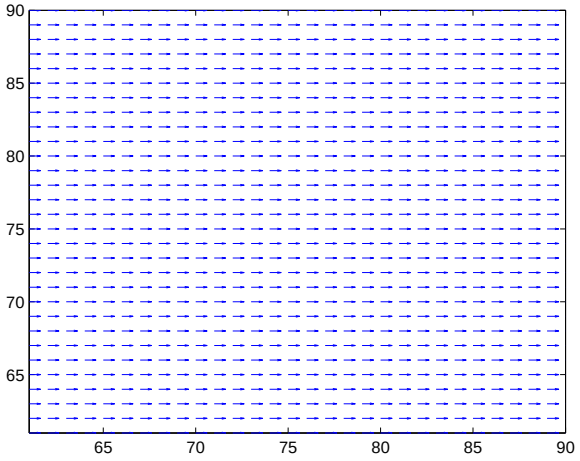
j) $\mathbf{P}(n) = \mathbf{P}^+(n) + \vec{g}^+(n) \cdot \vec{z}^{+T}(n)$

}

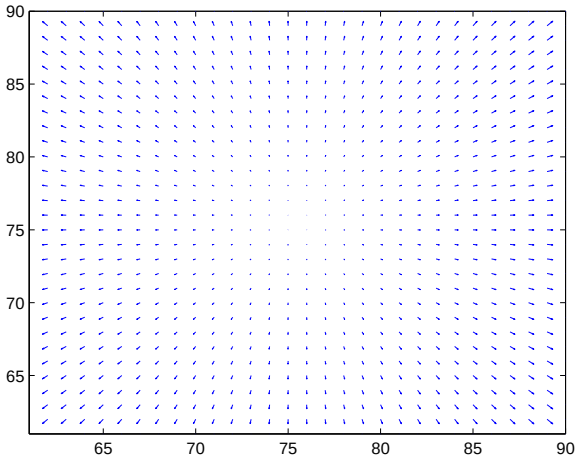
5.4 Experimental Results and Discussions

We performed simulations of our spatially recursive optical flow estimation algorithms based on two different frameworks, SR-OFE and RMR-OFE and the gradient-based optical flow estimation algorithm using a least-squares (LS) technique for constant and affine

(a)



(b)



(c)

Figure 5.4. The basic sample image and motion fields of translating and diverging image sequences.

motion models. Test image sequences are translational and diverging tree image sequences with size 150×150 in Fig. 5.4. A gaussian distribution function with $\sigma = 1.5$ is used for the 3-D smoothing kernel. For a derivative filter, we use Barron's 5-tap derivative filter [5]:

$$D = \frac{1}{12}(-1, 8, 0, -8, 1) \tag{5.31}$$

The performance of the algorithm is tested based on the angular error between ground truth and estimated motion. Table 5.3 shows the performance with thresholding. In addition, we perform simulations without thresholding to compare performance for the different number of updated pixels per LS computation among algorithms with 100% flow densities. Optical flow field results are presented in Table 5.5 for 100% flow densities.

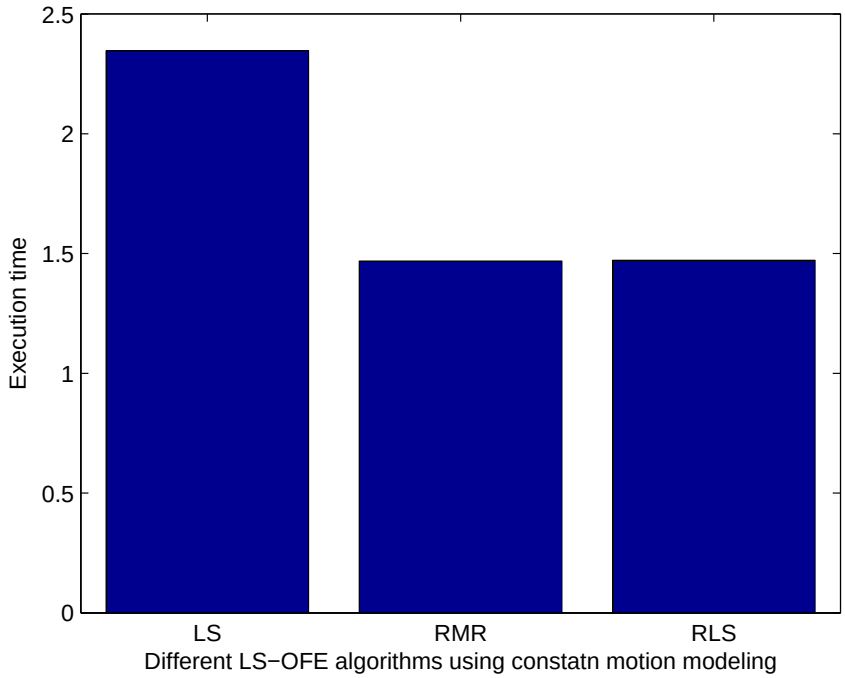


Figure 5.5. The execution speed comparison among LS-OFE, RMR-OFE, and SR-OFE. The parametric motion model is constant model.

For the constant motion model, the simulation results of the SR-OFE and RMR-OFE algorithm are the same as those of the LS-OFE algorithm for same window sizes and samples. However, there is some computational complexity reduction by changing the LS-OFE

Table 5.3. Angular error performance of spatially recursive LS-OFE algorithms with affine motion model. Thre means thresholding value. Ave and Std means average and standard deviation of angle error, respectively. Density indicates the optical flow density.

		Translating Tree			Diverging Tree		
No. of Updated Pixels	thre	Avg Error	Std	Density (%)	Avg Error	Std	Density (%)
3-pixels	0.01	0.87°	0.90°	31	1.90°	2.01°	48
2-pixels	0.01	0.80°	0.75°	20	1.83°	1.87°	37
1-pixels	0.01	0.76°	0.63°	9	1.64°	1.42°	20
3-pixels	0.001	1.08°	1.21°	68	2.10°	2.23°	78
2-pixels	0.001	1.10°	1, 25°	61	2.14°	2.28°	72
1-pixels	0.001	1.15°	1.39°	44	2.19°	2.34°	56

algorithm to the SR-OFE and/or RMR-OFE algorithm since the operations of generating LS matrix can be saved by data reuse (See Figure 5.5). Between the SR-OFE and RMR-OFE there is almost no difference in execution speed since the operations involving the 2×2 matrix for matrix inversion or system solution in Eq. 2.10 are simple to calculate.

Table 5.4. The parametric motion modelings of the constant and affine models for SR-OFE algorithms.

1. Constant motion modeling:
   $d(i) = -I_t(\vec{x}, t)$, $\vec{f^T}(i) = DM$, and $\vec{w} = \vec{p}$
   $\vec{I_s}(\vec{x}, t) = D$,
   $D = [I_x, I_y]$, $\quad M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\quad \vec{p} = [p_1, p_2]^T$
2. Affine motion modeling:
   $d(i) = -I_t(\vec{x}, t)$, $\vec{f^T}(i) = DM$, and $\vec{w} = \vec{p}$
   $\vec{I_s}(\vec{x}, t) = D$,
   $D = [I_x, I_y]$, $\quad M = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$,
   $\vec{p} = [p_1, p_2, p_3, p_4, p_5, p_6]^T$

For the affine motion model, the simulation results of the SR-OFE and LS-OFE algorithms are same if the number of data and windows for LS computation are same. The affine motion model [43] for SR-OFE is described in Table 5.4. The execution speed comparison of conventional LS-OFE and other fast spatially recursive LS-OFE algorithms is plotted in Fig. 5.6. The affine parametric motion model is employed, and codes are implemented in Matlab with highly optimized library for numerical linear algebra for Fig. 5.6.

In order to compare the number of operations of LS-OFE, SR-OFE, and RMR-OFE, first we convert least squares problems into normal equations (NE). In the SR-OFE, RLS with adaptive filtering are applied for the NE. Cholesky factorization is used to solve the NE for LS-OFE and RMR-OFE algorithm cases if the matrix is positive definite [54]. To save computations, some additional memory is used for the RMR-OFE and SR-OFE algorithms. By setting $N = 6$ for the affine motion model in Table 5.4, computational savings can be achieved for the RMR-OFE and SR-OFE (See Figure 5.6). The computational complexity of the SR-OFE is $O(kn^2)$, where k is the number of incoming point(s) per LS and n is the number of motion modeling parameters, and the RMR-OFE has $O(n^3)$ as the computational complexity. Therefore, the SR-OFE can be highly efficient if k is small enough compared with the number of parametric motion modeling number. This makes the SR-OFE to be most efficient when $k = 1$ in affine modeling (See Fig. 5.6). As k increases, the SR-OFE loses their advantage compared with the RMR-OFE. In fact, the execution speed of SR-OFE can be improved by using an efficient library for the inversion of symmetric matrices. Therefore, there is some redundancy in computing the inversion of symmetric matrix for SR-OFE.

The performance of the SR-OFE and RMR-OFE algorithms from 1 to 3 points update per LS operation are presented in Table 5.5. To compare the efficiency of the parametric motion models, the SR-OFE with affine motion model, ASR-OFE, is tested. The same derivative and smoothing kernels are employed for our simulations in Table 5.5 as described in [5]. Only gradient-based schemes with 100% flow densities are considered to compare global performance without thresholding. Based on Table 5.5, we can notice that the ASR-OFE has better performance for a smoothly spatial-varying motion field than a constant motion field. We also tested performance by using thresholding. It follows same tendency as Table 5.5 with improved performance.

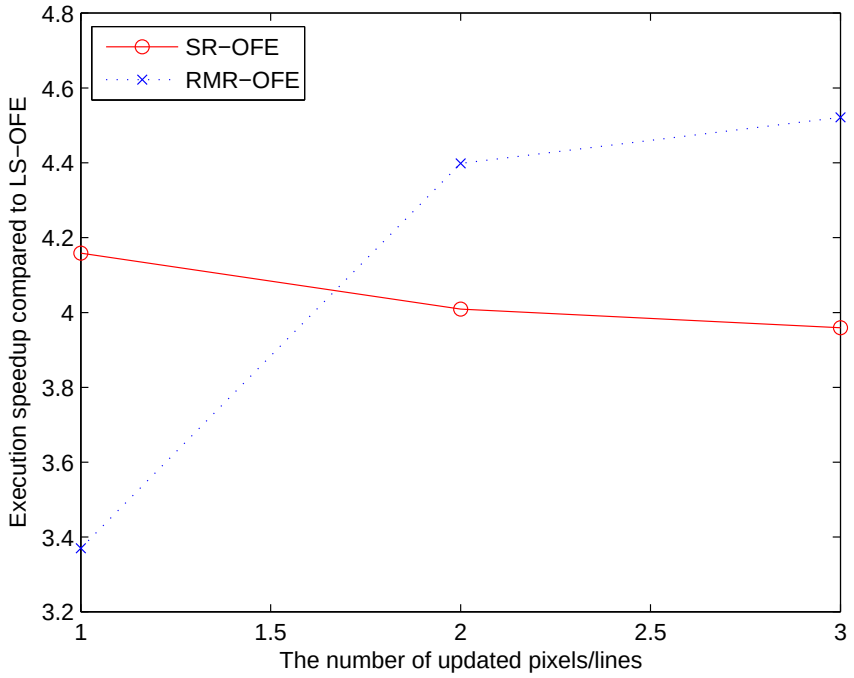


Figure 5.6. The execution speed-up comparison between RMR-OFE and SR-OFE normalized by the execution time of LS-OFE. The parametric motion model is affine model.

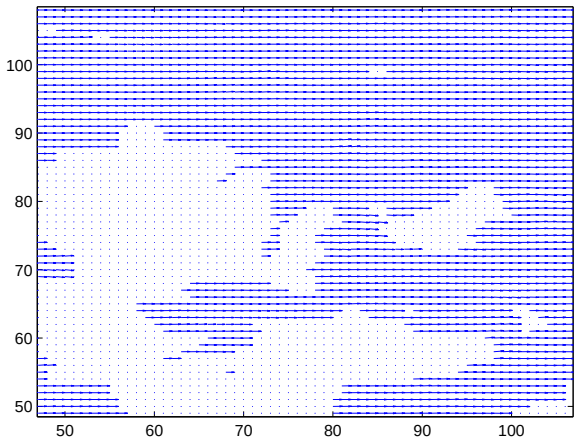
Table 5.5. Angular error performance of spatially recursive OFE with affine motion model. Density is dropped since every case has 100% flow densities. Avg and Std represent average and standard deviation, respectively.

	Translating Tree		Diverging Tree	
No. of Updated Pixels	Avg Error	Std	Avg Error	Std
3-pixels	1.34°	1.63°	2.14°	2.13°
2-pixels	1.42°	1.72°	2.22°	2.20°
1-pixels	1.68°	2.26°	2.50°	2.84°

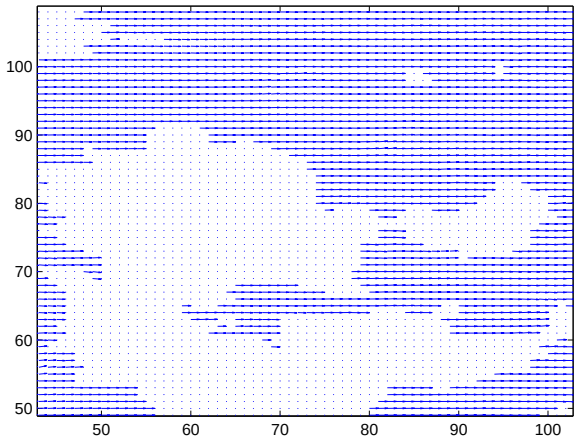
5.5 Conclusions

In this chapter, we propose a spatially recursive optical flow estimation (SR-OFE) framework using adaptive filtering. The sliding window RLS and adaptive filtering are employed to reduce the redundancies for calculating successive LS among adjacent pixels. In addition, we also introduce a LS-OFE framework using recursive matrix refinement (RMR-OFE). Even though we describe constant and affine motion models, this framework can be applied to other parametric motion models. We have been able to find parametric motion

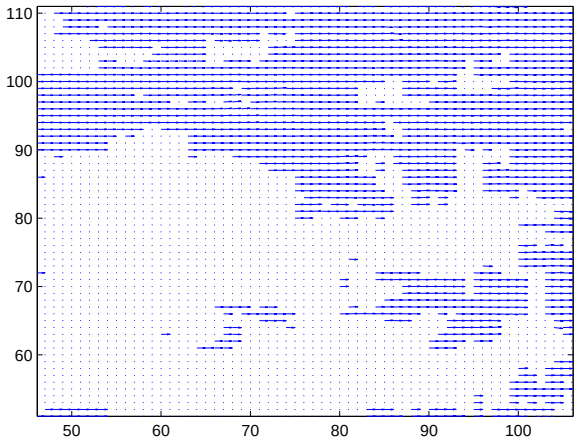
models with up to 12 parameters [6]. In addition, some algorithms such as ego-motion estimation, multi-frame OFE, and 3-D OFE, need more number of motion parameters for estimating than general 2-D OFE. Based on these results, we can conclude that our algorithm frameworks can improve the saving of computational complexity more as the number of motion modeling parameters increases because RMR-OFE efficiently reuse the data of linear system matrix and standard LS needs $O(n^3)$ and RLS requires $O(kn^2)$ computations, where n is the number of motion modeling parameters and k is the number of updated inputs per LS computation.



(a)

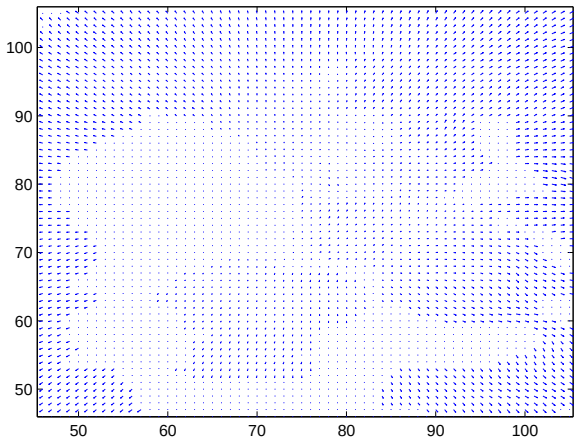


(b)

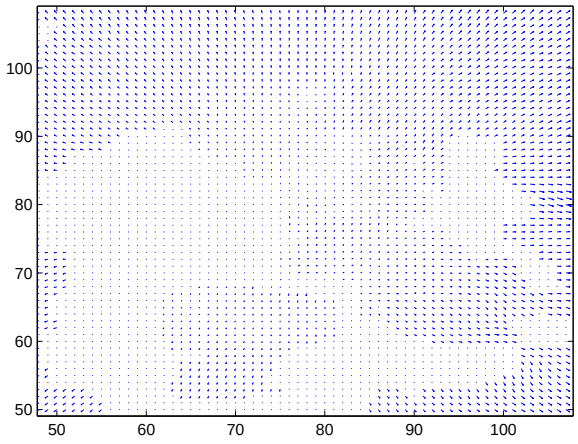


(c)

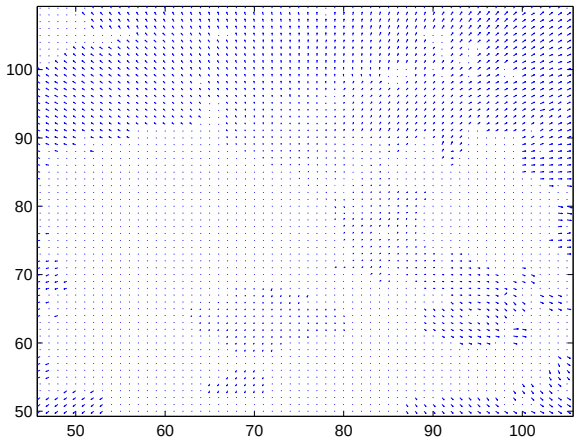
Figure 5.7. The optical flow fields for different number of pixel update per LS computation with threshold = 0.001. The parametric motion model is affine model. The test sequence is translational tree image. (a) 3-pixel update case; (2) 3-pixel update case; (c) 1-pixel update case



(a)

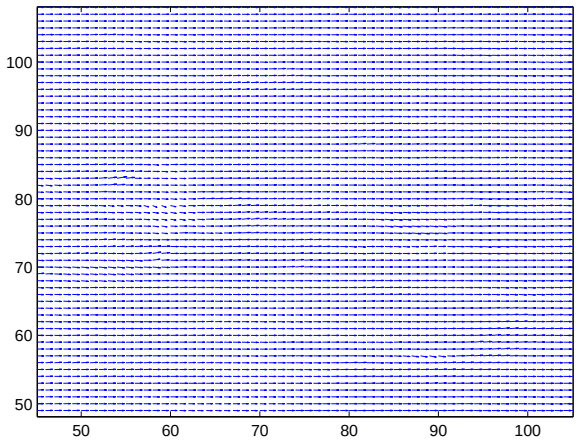


(b)

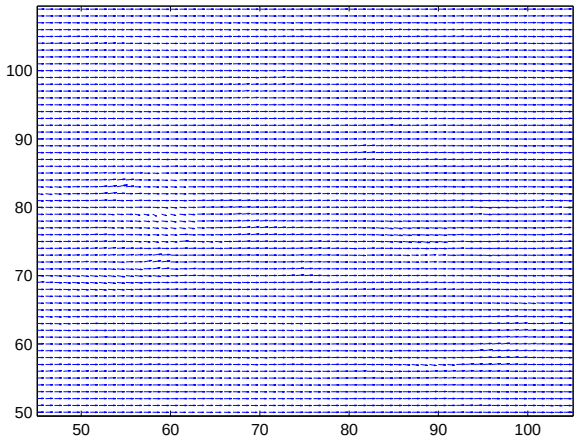


(c)

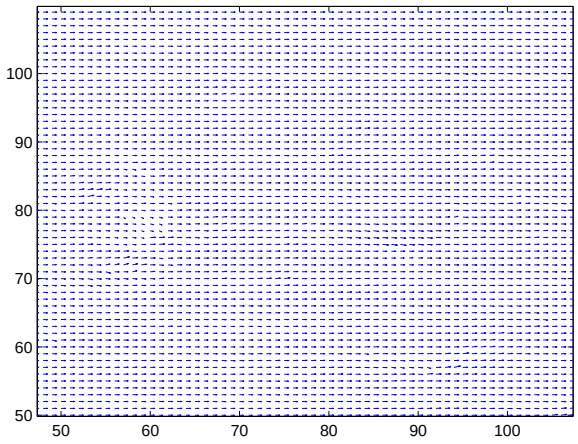
Figure 5.8. The optical flow fields for different number of pixel update per LS computation with threshold = 0.001. The parametric motion model is affine model. The test sequence is diverging tree image. (a) 3-pixel update case; (2) 3-pixel update case; (c) 1-pixel update case



(a)

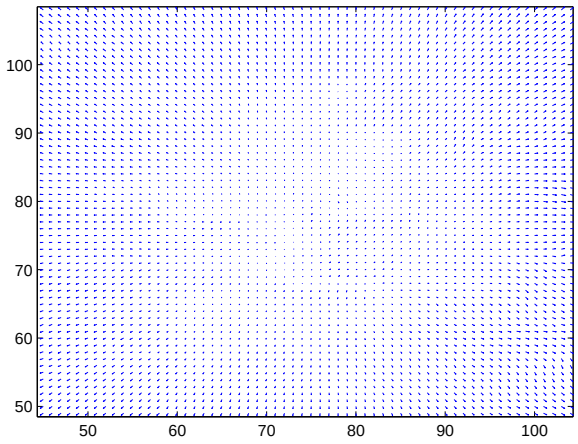


(b)

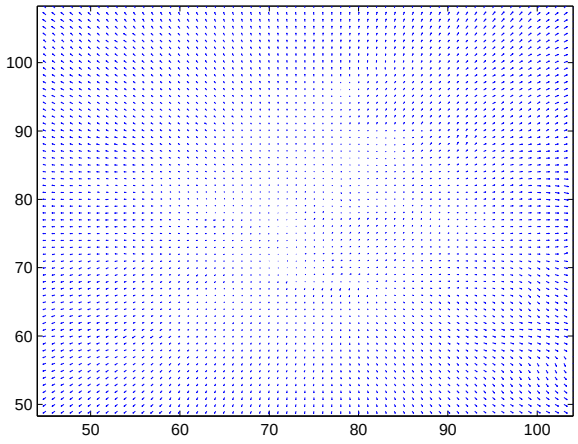


(c)

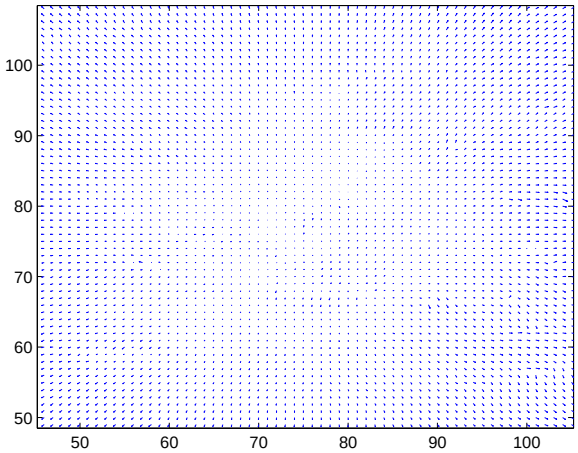
Figure 5.9. The optical flow fields for different number of pixel update per LS computation. The parametric motion model is affine model. The test sequence is translational tree image. (a) 3-pixel update case; (2) 3-pixel update case; (c) 1-pixel update case



(a)



(b)



(c)

Figure 5.10. The optical flow fields for different number of pixel update per LS computation. The parametric motion model is affine model. The test sequence is diverging tree image. (a) 3-pixel update case; (2) 3-pixel update case; (c) 1-pixel update case

CHAPTER 6

THE MULTI-RESOLUTION MOTION ESTIMATION USING TEMPORAL ALIASING DETECTION

In this chapter, we suggest a new algorithm for wavelet-based multi-resolution motion estimation (MRME) using temporal aliasing detection (TAD). This method improves rate-distortion performance between motion compensated and current images compared to conventional MRME and MRMEs with motion vector (MV) prediction through median filtering. Originally this idea is inspired by the multi-dimensional filtering for motion estimation, which is introduced in the background chapter of this dissertation.

Wavelet-based MRME can be implementable by using the STI since the motion estimation is performed in frequency-domain and 2-D FIR filterings are required for wavelet transform. Therefore, this can be another motion estimation system using the CADSP approach.

This chpater is organized as follows. A brief introduction of conventional MRME is presented in section 6.1. The temporal aliasing cases in the wavelet transform are described in section 6.2. The proposed MRME-TAD algorithm is explained in section 6.3. In section 6.4, we show and compare the performance of the proposed algorithm with conventional MRME and MRMEs with MV-prediction through median filtering and conclusions are presented in section 6.5. Finally conclusions are described in sectino 6.6.

6.1 Introduction

The discrete wavelet transform (DWT) has emerged as a promising technique for image processing applications such as image/video coding and denoising. The primary reasons are its flexibility in representing nonstationary image signals and its good match to the human visual systems. The wavelet representation provides a multiresolution/multifrequency expression of a signal with localization in both time and frequency. Its properties make it

particularly useful in image and video coding and other application. For tutorials on the wavelet transform and its applications to image/video coding, some information can be found in [72, 66, 59, 62, 47, 70]

In video coding, some type of interframe prediction is often used to reduce the interframe redundancy. Motion-compensated prediction has been used as an efficient scheme for temporal prediction. In order to perform the motion compensation in the wavelet domain, block matching motion estimation has been applied, resulting in multiresolution techniques for motion estimation [81, 34, 52]. The multi-resolution motion estimation (MRME) can be categorized as a hierarchical block motion estimation (HBME) approach using variable block sizes. Since MRME estimates motion vectors (MVs) hierarchically, it can reduce the computational complexity and achieve high performance by exploiting the correlation among the wavelet subbands. The multi-resolution motion estimation (MRME) can be applied for scalable video coding seamlessly [34].

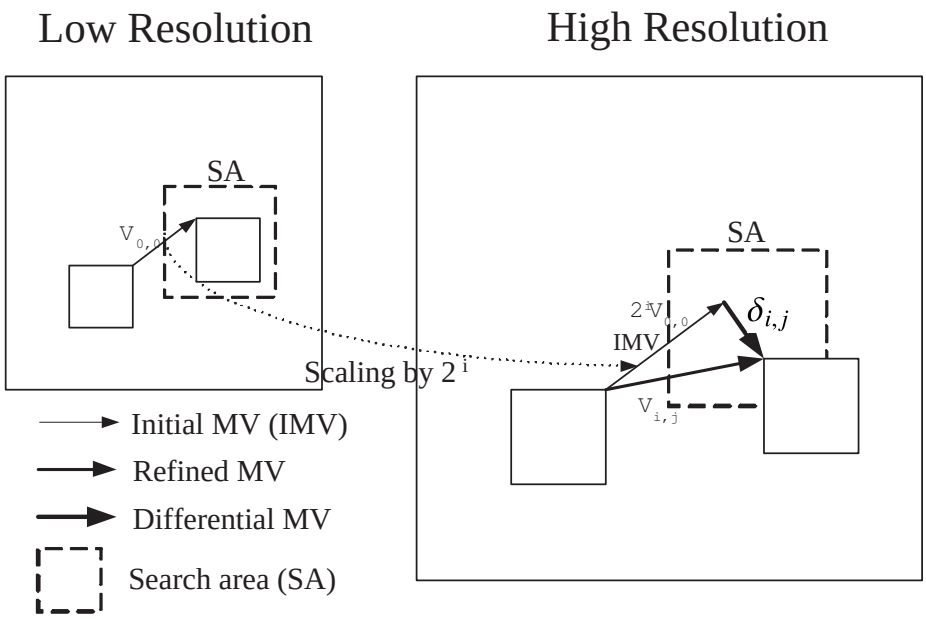


Figure 6.1. Hierarchical multi-resolution motion estimation using variable block sizes.

In conventional wavelet-based MRME schemes, motion vectors (MVs) are first estimated at the lowest resolution. This is reasonable since most of the image energy is

preserved at this resolution. And then MV estimates are refined at higher resolutions depending on the corresponding initial MVs at lower resolutions to achieve high performance and exploit cross correlations in a wavelet transformed image (see Fig. 6.1). Each lower resolution image is subsampled by two from the previous higher resolution image. This makes the number of MV blocks equal in each subband because of the pyramidal data structure in a tree-structured wavelet transform.

In spite of these advantages, the wavelet-based MRME has some motion-related aliasing problems. One problem is caused mainly by the downsampling operation, which is spatially-variant [69]. Therefore, direct motion estimation in the wavelet domain is subject to aliasing. Another problem arises from the velocity of object. In subband decomposition, temporal aliasing effects increase as the motion of the object increases, causing the performance of the conventional MRME algorithms to drop. The worst-case aliasing results in the sign change, which means the opposite direction to the true motion in motion estimation. In this study, we investigate how to address these phenomenons without changing original signal.

Previous works regarding algorithms for MRME and reducing aliasing are reported in [33, 80, 78, 52]. Kim, *et al*, [33] presented a two-stage variable block size MRME algorithm to improve coding gain by grouping similar MVs. A bottom-up construction of a quadtree is employed for grouping. Zan, *et al*, [80] proposed MV prediction techniques to reduce the propagation of the false MVs. Among three different techniques, they found median filtering to be the best considering coding gain, the number of bits, and computational complexity. Yang and Ramchandran [78] suggested an interpolation method to reduce spactial-variant aliasing by FIR filtering. However, this needs some training to find optimal values and the low band images must be doubled by interpolation. In addition, the motion estimation scheme is only applied to low frequency bands. Park and Kim [52] used a low-band-shift method to overcome space-variant aliasing. Even though it outperforms conventional MRME in terms of PSNR, it loses the advantage of scalability by reorganizing

the wavelet blocks.

6.2 Conventional Multi-Resolution Motion Estimation (MRME) and Median Filtering Technique

In this section, we will review conventional MRME based on a three-level wavelet analysis shown in Fig. 6.2, which is defacto standard for conventional MRME.

Conventional MRME has been developed to reduce computational complexity and achieve high performance by exploiting the correlation among the wavelet subbands. Originally four algorithms were proposed by Zhang and Zafar [81]. Those schemes are classified depending on the different approaches of reference MV estimation in lowest resolution (0^{th} level in Fig. [81]) and MV refinement in high and mid frequency subbands.

Among four different algorithms, *Scheme-III*, $S_8 + refine$ in [81], is the best overall performance in terms of rate-distortion. This scheme will be the subject of further investigation in this paper.

In *Scheme-III*, MVs are first estimated for the low frequency subband in the lowest resolution, i.e., *LL0*, and are properly scaled depending on the resolution and used as reference MVs of further refinement for all other subbands (see Fig. 6.1). The process can be expressed as

$$V_{i,j}(x,y) = 2^i V_{0,0}(x,y) + \delta_{i,j}(x,y) \quad for \quad i = 0,1,2 \quad and \quad j = 1,2,3, \tag{6.1}$$

where $V_{i,j}$ is the MV at the resolution level i with orientation j and $\delta_{i,j}$ is a refinement term due to the search around the predicted value $2^i V_{0,0}$. Three different values are used for i because we only consider three-level wavelet transformed images.

A serious drawback with the conventional MRME algorithms in [81] is the propagation of false MV. This is caused by the prediction step from MVs at the lowest resolution. Whenever false MV is used as a basis MV for MV prediction at higher resolutions, there is no way to prevent the false MV from propagating into other higher frequency subbands.

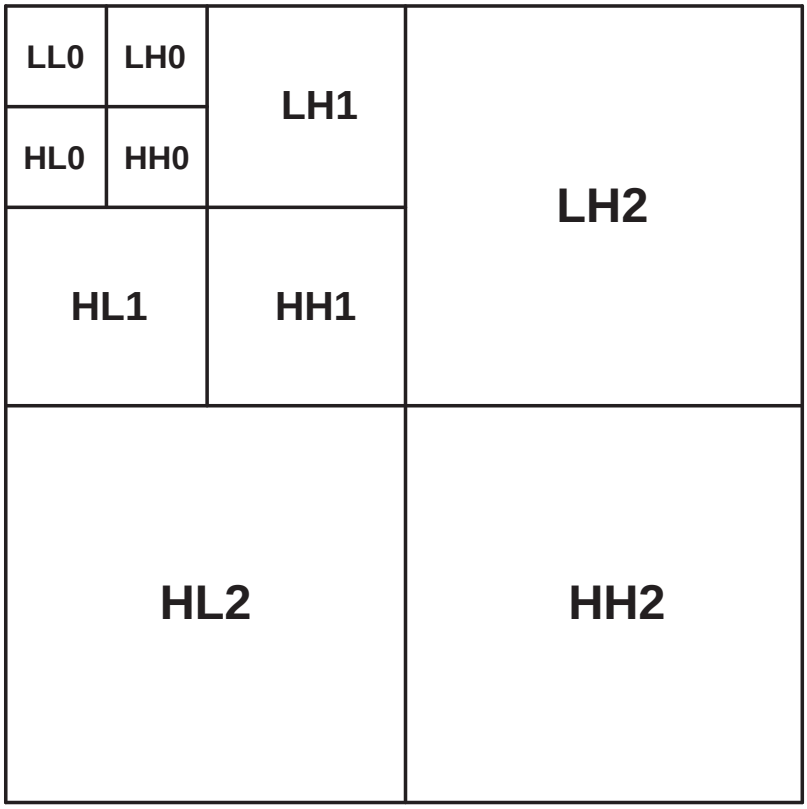


Figure 6.2. Frequency decomposition of an image. LL represents the lowest frequency subband. LH and HL means horizontal and vertical edge subbands respectively. HH indicats the diagonal edge subbands. The number after L or H letter shows the wavelet transform level.

In order to improve this situation, some techniques are suggested in [80]. Considering the computational complexity and performance, the best choice is the MV prediction through median filtering. This approach basically uses the statistical property of asymptotic gaussian pdf for MV at video signal under noise. By using median filtering, the best estimate can be obtained since median and mean are same for Gaussian pdf.

To perform the MV prediction through median filtering, first the basis MV for prediction should be estimated at the lowest resolution band. In *Scheme-III*, the basis MV is estimated in *LL*0. For other subbands except *LL*0, the basis MV after median filtering will be used as the predicted MV. To perform the median filtering of basis MV, the candidate MVs are composed of the MVs of same and neighboring block positions at *LL*0. Then horizontal and vertical components of candidate MVs are median-filtered separately.

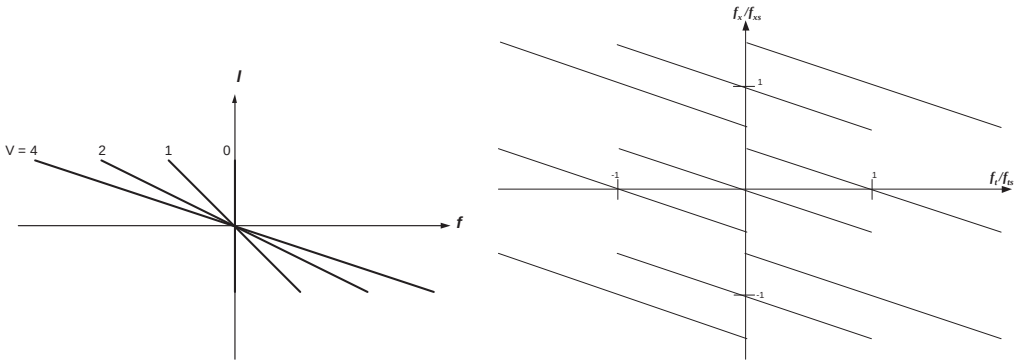


Figure 6.3. Spatio-temporal spectrum based on the constant motion model.

This median filtered MV will be used as a prediction MV for motion estimation at other subbands.

6.3 Aliasing of the Wavelet Domain in a Video Signal

We will discuss about two types of aliasing by the motion in wavelet transformed video signals in this section. First the temporal aliasing is caused by the discrete-time undersampling compared to the speed of motion. Another part is the aliasing caused by the downsampling. These phenomenons will be described and discussed when their characteristics respectively.

6.3.1 Temporal Aliasing by High Speed Motion

Typically block matching algorithms assume block-wise smooth and constant motion among successive frames. For a one-dimensional time-varying image $i(x,t)$ moving with constant velocity v_x, this assumption can be modeled by

$$i(x,t) = i(x - v_x t). \tag{6.2}$$

After taking the Fourier transform, we obtain

$$I(f_x, f_t) = I(f_x)\delta(v f_x + f_t), \tag{6.3}$$

where $\delta(vf_x + f_t)$ is the 1-D Dirac delta function and $I(f_x)$ is the 1-D transform of the stationary image $i(x)$. Thus, it is clear that the energy of $I(f_x, f_t)$ is only on the line $f_t = -vI_x$ (see Fig. 6.3(a)).

If the image is sampled in the time and spatial domains with sampling frequencies f_t and f_x respectively, the spectrum of $I(f_x, f_t)$ in Fig. 6.3(a) will be replicated in the f_t and f_x directions at the intervals of f_{xs} and f_{ts} respectively. This is shown in Fig. 6.3(b) for a motion of 4 pixels/frame. According to Fig. 6.3, temporal aliasing will be more severe as velocity increases [12] and is hard to be removed by temporal lowpass filtering as an anti-aliasing filter because the number of frames in camera device or photo sensors is fixed based on system requirements or video coding standards.

The temporal spectrum at a particular position x_o can be described by the 1-D Fourier transform:

$$I_{x_o}(f_t) = \frac{1}{|v|} \sum_{k=-\lfloor |v| \rfloor}^{\lfloor |v| \rfloor} I\left(-\frac{f_t + k}{v}\right) e^{-j2\pi(f_t - k)x_o/v}, \tag{6.4}$$

where $\lfloor |v| \rfloor$ is the largest integer less than $|v|$. The derivation of Eq. (6.4) is described in Appendix I. This equation illustrats the temporal aliasing by the overlap of frequency responses with respect to the f_t axis when the projection of Fig. 6.3(b) into the f_t axis is happened. This aliasing can be regarded as temporal downsampling effect with linear phase modulation. The downsampling rate is determined by the speed of object and camera. Therefore, this aliasing is unavoidable for any temporal filtering operation if v is greater than one pixel/frame, and will be worse as v increases.

However, we can reduce the temporal aliasing phenomenon by using spatial lowpass filtering. Based on these observations of temporal aliasing for high velocity spatial-temporal signal such as in Fig. 6.4, we can regard the low frequency subband at the lowest resolution (LL) in Fig. 6.4 as being safe from temporal aliasing caused by high speed motion. However, other higher frequency subbands can suffer from temporal aliasing. This will deteriorate the accuracy of the motion estimation.

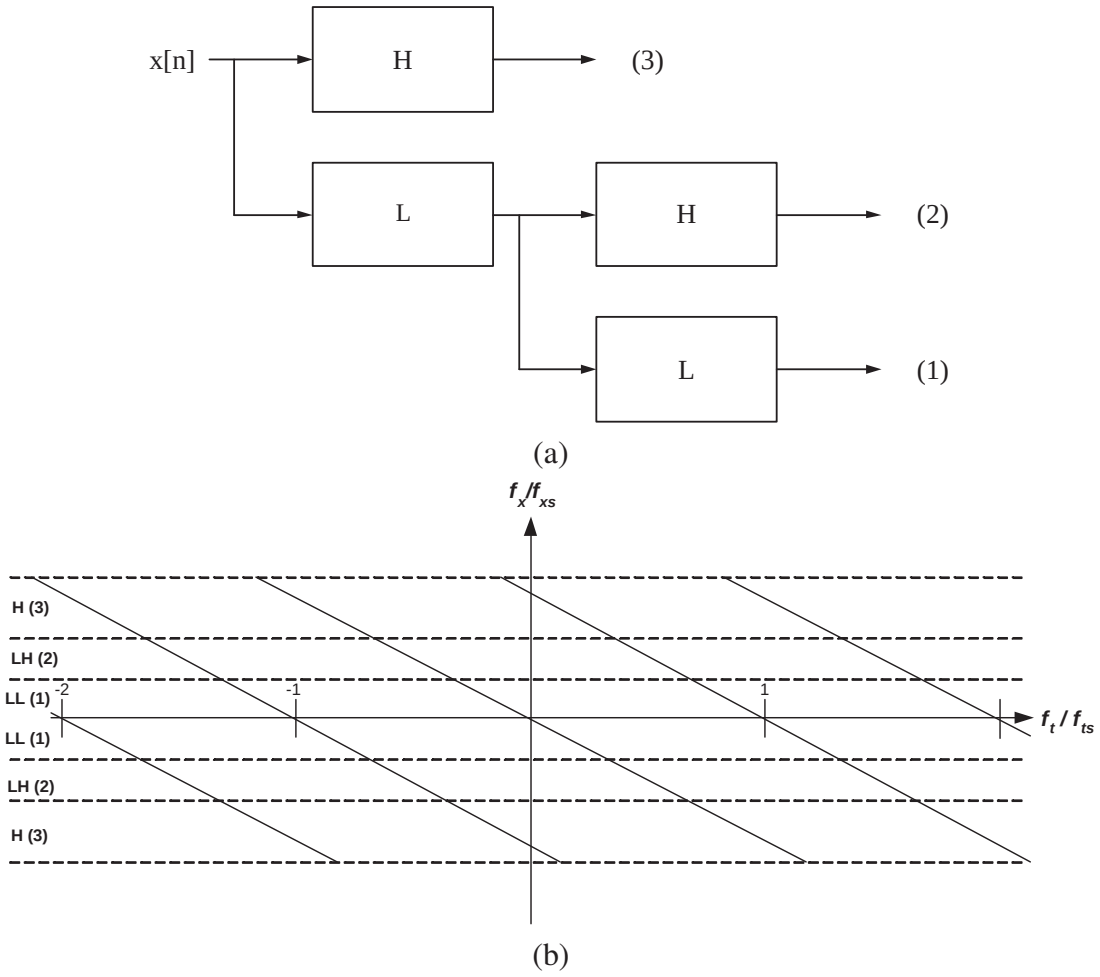


Figure 6.4. Example of the efficiency of low-pass filtering for the video signal under temporal aliasing by high speed motion. (a)one-dimensional 2-level subband filtering; (b)Frequency decomposition by subband filtering of (a) in temporal-spatial spectrum. H is the high-pass filtering, LH is the low-pass filtering followed by high-pass filtering, and LL is the low-pass filtering followed by low-pass filtering.

If MRME is based on undecimated DWT or have negligible aliasing by spatial down-sampling, we can regard the low frequency subband at the lowest resolution (*LL0*) in MRME as *LL* band in Fig. 6.4. This assumption can be more reasonable as more refined sub-pel is employed (i.e. half or quarter-pel) because interpolation can reduce aliasing by spatial downsampling. By using downsampling for each wavelet filtering in MRME in Fig. 6.4, the effect of high speed would be more diminished as the resolution is reduced.

6.3.2 Temporal Aliasing by the Spatially-Variant Property

In addition to the temporal aliasing by the high v, there is another temporal aliasing caused by the downsampling in multirate filtering. In discrete-time spatial samples, the equations in Eq. 6.2 can be described as follows under translational motion:

$$x[n] = i[n, t]; \quad y[n] = i[n, t + 1], \tag{6.5}$$

$$y[n] = x[n - v], \tag{6.6}$$

where $x[n]$ is the current image, $y[n]$ is the same image as $x[n]$ with motion vector v.

If we perform filtering $H(w)$ for x[n] and y[n] with downsampling, the filtered result will be

$$X_1(w) = \frac{1}{2}\left(H\left(\frac{w}{2}\right)X\left(\frac{w}{2}\right) + H\left(\frac{w}{2} + \pi\right)X\left(\frac{w}{2} + \pi\right)\right), \tag{6.7}$$

and

$$Y_1(w) = \frac{1}{2}\left(H\left(\frac{w}{2}\right)Y\left(\frac{w}{2}\right) + H\left(\frac{w}{2} + \pi\right)Y\left(\frac{w}{2} + \pi\right)\right). \tag{6.8}$$

The Fourier transform of Eq. (6.6) is

$$Y(w) = X(w)e^{-jwv}. \tag{6.9}$$

Substitution of Eq. (6.8) with Eq. (6.9) leads to

$$Y_1(w) = \frac{1}{2}\left(H\left(\frac{w}{2}\right)X\left(\frac{w}{2}\right)e^{-j(w/2)v} + H\left(\frac{w}{2} + \pi\right)X\left(\frac{w}{2} + \pi\right)e^{-j(w/2+\pi)v}\right). \tag{6.10}$$

Both of Eq (6.7) and (6.8) have aliasing parts. If we consider the ideal motion vector in the down-sampled signal, $v_1 = v/2$, where v_1 is the down-sampled version of the original ideal motion vector. Based on Eq. (6.9), we expect

$$Y_1(w) = X_1(w)e^{-jw(v/2)}. \tag{6.11}$$

However, Eq. (6.11) is not satisfied because of aliasing terms in Eq (6.7) and (6.8). Therefore, conventional MRME can experience aliasing problems because of downsampling effect.

6.4 Multi-Resolution Motion Estimation (MRME) using Temporal Aliasing Detection (TAD)

In this section, we propose a new algorithm for MRME to exploit the aliasing phenomena in wavelet-based MRME. Compared to the conventional MRME, this algorithm has additional steps such as temporal aliasing detection step for MRME and post-processing of mode map and MV.

6.4.1 Temporal Aliasing Detection Step for MRME

From the previous section, we notice that two types of temporal aliasing exist in conventional MRME. If the motion of the video signal can be well modeled by Eq. (6.2), we can assume that the motion in the *LL* band of Fig. 6.4 will be small enough to neglect the aliasing effect by high speed. The temporal aliasing by spatial variant property can be improved by interpolation. These lead *LL0* in MRME to be little affected by temporal aliasing if the wavelet transform is three or more levels deep as illustrated in Fig. 6.2 and refined sub-pel ME is employed. These facts allow us to assume that the MV from *LL0* band of conventional MRME can be used for reference MV for other higher frequency bands.

Since the *LL0* band is little affected by aliasing based on our assumptions, MVs (motion vectors) in *LL0* can be used as reference MVs for the motion estimation in other high frequency subbands in conventional MRME algorithms. However, the other mid- and high-pass subbands can experience two different aliasing effects because the absolute difference operation in motion estimation algorithms is a magnitude response of the temporal difference filtering. As the velocity of an object in a sequence of images increases, temporal aliasing can be severe. The worst response of aliasing is backward motion caused by frequency folding or sign change [51]. This phenomenon has been reported in [64] and [41]

and we also have noticed that by watching car advertisements on television, which is called the "Western Wheel Effect" [41]. The backward motion means that there is probability that spatial area in high frequency subbands, which are affected by temporal aliasings, has correct MV around the location of the opposite direction of the reference MV. By finding the refined MV that has the smaller error between the refined MVs from the same and the opposite locations of the reference MV of *LL*0, we can decide in which wavelet subbands temporal aliasing is severe (see Fig. 6.5). We call this as the temporal aliasing detection (TAD) step.

By using TAD, we can detect whether the current motion block in a specific wavelet transform level is under some temporal aliasing or not. If we find that the current block is affected by temporal aliasing, a special symbol should be coded to flag. This special symbol can be regarded as mode selection between normal and temporal aliasing modes for the interpretation of the reference MV. Other steps follows conventional MRME schemes.

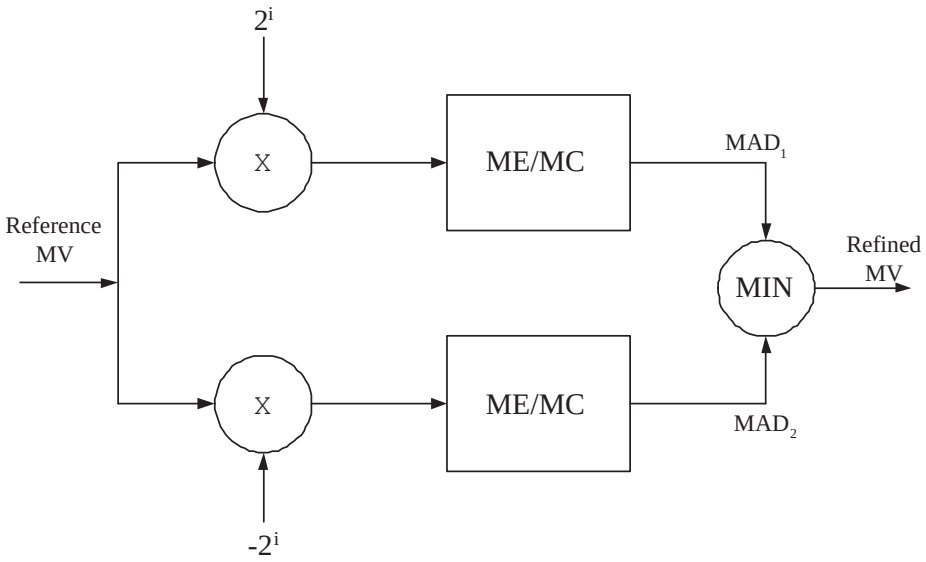


Figure 6.5. Block diagram for the multi-resolution motion estimation using temporal aliasing detection (MRME-TAD). *i* is integer value depending on the wavelet level.

6.4.2 Post-Processing of Mode Map and MV

From our experiments, the result of MRME-TAD generates aliasing and conventional blocks with misclassification. This can be caused by noise, motion model mismatch, and other aliasing effects. Even though the aliasing and conventional mode decision is correct, it could be unhelpful if very low improvement of MAD performance is achieved. To enhance rate-distortion performance, we need to perform post processing for mode map and MV by exploiting temporal aliasing properties. This is based on inter-band relationship for temporal aliasing by high speed similar to Fig. 6.4, which is possible when sub-pel ME is employed to reduce aliasing by down-sampling. We expect the mid-frequency bands are less affected by aliasing than high frequency. It leads the highest frequency band to be most aliasing-prone band and the aliasing-prone property would be reduced as the frequency of subband moves toward mid- and low-frequency regions. Temporal aliasing by downsampling is also considered by allowing the unexpected aliasing mode in mid-frequency band without an aliasing mode in high-frequency bands. The procedure for post-processing of mode map is described as follows:

Case 1: For the same block position (x, y) of all high-frequency bands with same directionality, if there is one aliasing block, we expect it is happened at highest frequency band based on Fig. 6.4. However, there are some aliasing motion blocks not at the highest frequency band, but other high frequency bands. For those subbands, if the difference of minimum absolute difference (MAD) between aliasing and conventional mode is less than threshold, it will be defined as non-aliasing mode and use conventional MV as reference MV. Otherwise, the original mode after MRME-TAD will be kept.

Case 2: For the same block position (x, y) of all high-frequency bands with same directionality, if there is two or more aliasing block, we expect it is happened from highest to other successive high frequency bands based on Fig. 6.4. If it is not, we expect all high frequency band will be turned into aliasing mode for the same direction. Therefore, if the difference of MAD between aliasing and conventional mode is less than threshold, it

will be defined as aliasing mode and use aliasing MV. Otherwise, the original mode after MRME-TAD will be kept.

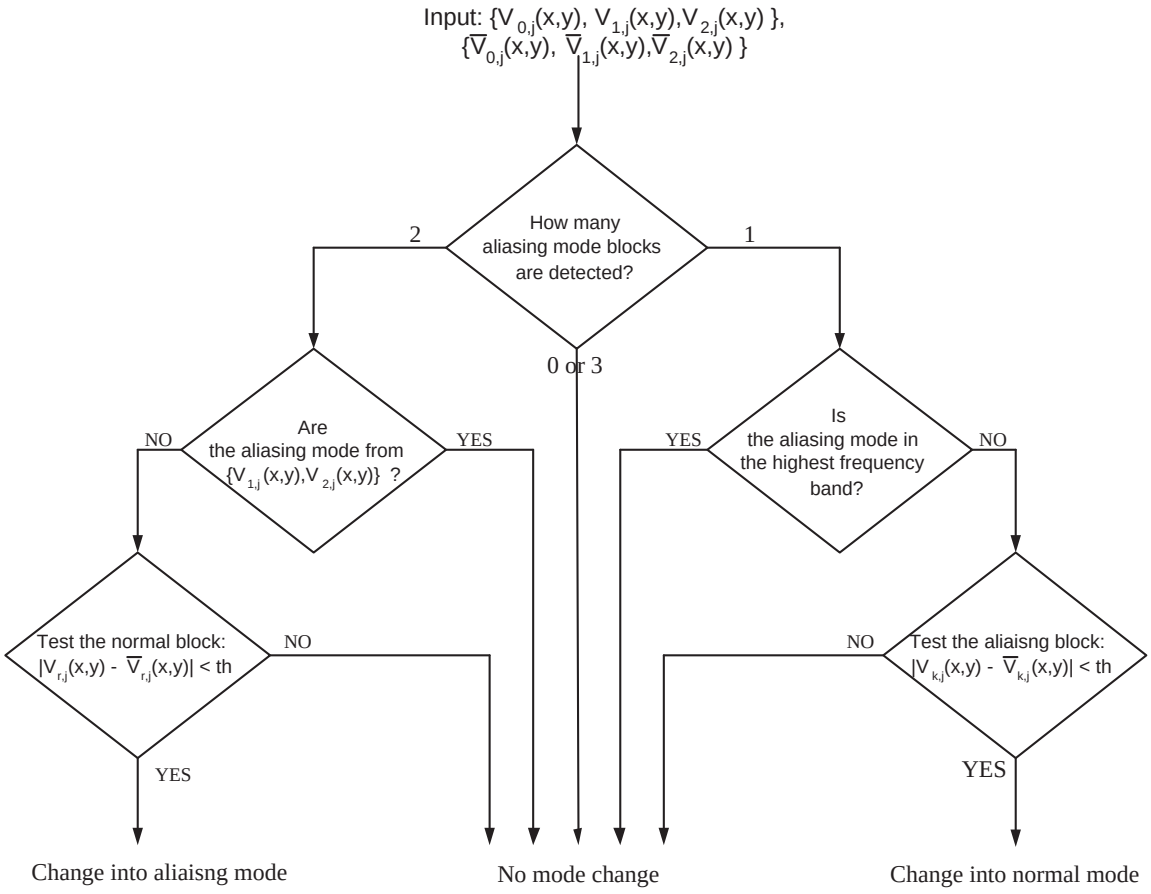


Figure 6.6. Flow chart for the post processing of block mode and MV information. As input parameters, $V_{i,j}$ is the normal mode MV at the resolution level i with orientation j and $\overline{V}_{i,j}$ the aliasing mode MV at the resolution level i with orientation j. Three different values are used for i because we only consider three-level wavelet transformed images. Th is threshold for mode change decision. r in $|V_{r,j} - \overline{V}_{r,j}|$ means the normal mode resolution except two aliasing mode blocks and k in $|V_{k,j} - \overline{V}_{k,j}|$ means the aliasing mode resolution except two conventional mode blocks.

This procedure is described in Fig. 6.6. In this post processing, we only use one threshold parameter as error tolerance for aliasing and conventional mode. After finishing post processing for mode map information, we perform quadtree coding for mode map (Fig. 6.7). In quadtree coding, '0' is assigned for normal mode and '1' is used to indicate aliasing mode. The node of quadtree is divided more detail if there is aliasing mode ('1') within the node. Otherwise, the node of quadtree is indexed as '0' and no more bit coding

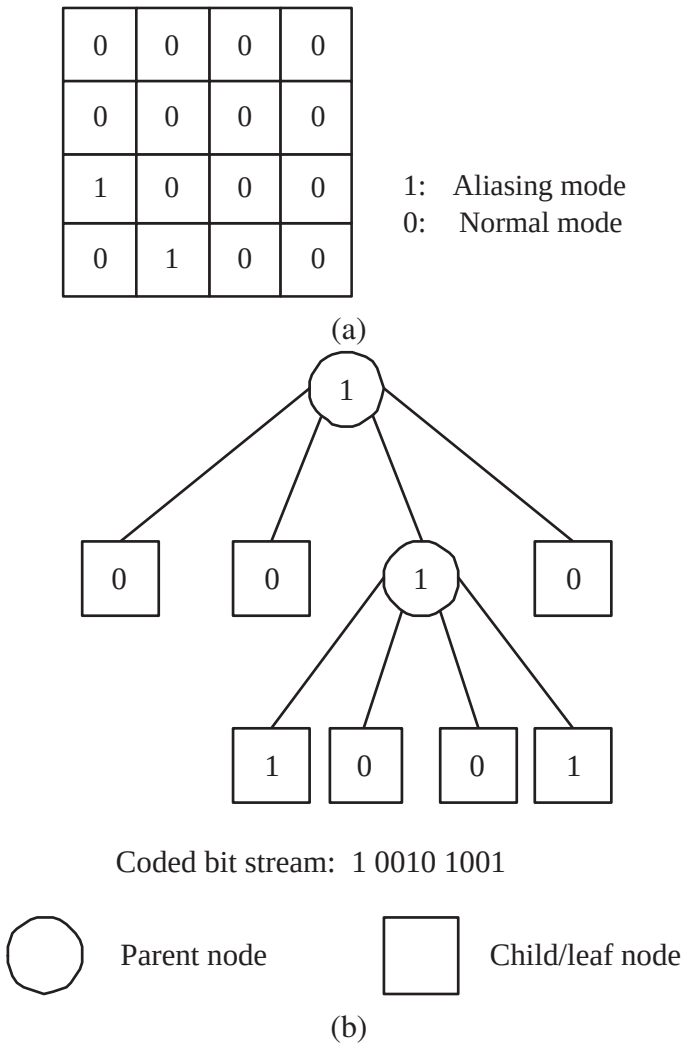


Figure 6.7. Example of quadtree coding process for mode map coding. (a) sample mode map pattern; (b) quadtree map for (a).

is necessary. The subblocks within a parent node region are scanned as the order of z-scan. The worst case rate-distortion cost by adding quadtree for the non-aliasing-mode subband would be 1-bit ('0'), which is negligible, without any distortion reduction. Therefore, the added quadtree bits for mode map can be profitable if it effectively capture aliasing mode information.

6.4.3 Mathematical Performance Analysis for the Temporal Aliasing by Downsampling

Even though the temporal aliasing detection is mainly for the heavy temporal aliasing by the temporal deficient sampling of the fast motion, this idea can help to reduce the temporal aliasing for sufficient temporal sampling case, *i.e.* $v <= 1$. In other words, the aliasing is caused by the spatial variant property.

The detailed derivation is described in Appendix II. Based on our derivation, we conclude that the aliasing can be reduced by using temporal aliasing detection step in the odd-pixel movement case.

6.5 Experimental Results and Discussions

This section presents the simulation results and discussion about performance tests. The first test is for whether our temporal aliasing model is reasonable or not by using translational dominant image sequence. Other tests are the rate-distortion performance of MPEG test sequences with our proposed algorithm and other standard/state-of-the-art algorithms in wavelet-based MRME. For all experiments we use biorthogonal 2/6 wavelet to generate a three-level wavelet tree image. In MRME and MRME-TAD algorithms, the block size at the lowest resolution level is 2×2. In addition, all MVs are DPCM encoded with respect to their predictions except for the MVs in the *LL*0 band, where they are DPCM encoded on a row-by-row basis and then arithmetic coding with adaptive modeling is employed [76]. As described in section 6.4, we assign a symbol for MRME-TAD to notify the reference MV interpretation change as aliasing and normal modes using quadtree coding. As a metric for distortion, normalized predictive mean square error (NPMSE) is defined as the ratio of the sum of squared errors between correct and motion-estimated frames and the total number of pixels of original image size. The reason why we employ this metric instead of pure mean-square error (PMSE) is that we would like to compare average predicted MSE of subbands for distortion and the number of bits for rate. However, tree structured wavelet transform makes the size of subbands different per level. In order to compare fairly, we

use the total number of original image pixels as the normalization factor. As MPEG test sequence, we use *flower garden*, *football*, and *mobile and calendar*.

In the first test, MRME-TAD algorithm is used for globally horizontally translational moving images to validate whether our temporal aliasing model is reasonable or not. As a test sequence, we select *flower garden* image sequence with SIF format (See Fig. 6.8). Table 6.1 shows the results of the number of aliased and non-aliased blocks based on TAD

Table 6.1. The number of aliased and non-aliased blocks for the *flower garden* image sequence. The total number of motion blocks is 2970.

Number of Blocks	Current Frame Number (Reference frame number: 55)					
	56	57	58	59	60	61
Normal	2759	2605	2507	2476	2319	2356
Aliased	211	365	463	494	651	614

step. As the frame number of the next frame increases, the motion becomes larger. Based on our model in Eq. (6.2), temporal aliasing components play an important role as motion becomes large. Overall results of Table 6.1 ratify what we described in section 3.

We tested MRME-TAD for MPEG test sequences with SIF formt. Figure 6.9. shows that the rate-distortion performance as the search range increases from 3×3 to 11×11 for *Football* sequence (See Fig. 6.10). The metric of rate is the average number of bytes for MV of subbands and the metric of distiortion is the average NPMSE of motion compensated images. Overall MRME-TAD with median filtering is the best performance. In Fig. 6.9, we can notice that additional TAD method beats every non-TAD methods for *Football* test sequence. One of reasons why TAD make better performance is that *Football* test sequence has fast and dynamic motions. This makes TAD methods have better performance than non-TAD methods because fast and dynamic motions generate the higher chance of aliasing. Figure 6.12 shows that result of *Mobile and Calendar* test sequence (See Fig. 6.11). This test sequence is a globally moving sequence. Even though MRME+TAD gives better performance than MRME, there is little difference between MRME+median

and MRME+TAD+median. This is caused by median filtering property. For globally moving sequence case, the distribution of MV will be locally Gaussian distributed. Typically median filtering has same effect as the best linear estimator under additive Gaussian distribution noise [31]. Since median estimator effectively reduces aliasing noise effects, the additional gain of TAD can be small.

The computational complexity of MRME-TAD is higher than that of conventional MRME. Typically, rate-distortion is more important than computational complexity in video coding. However, we can trade-off the performance and computational complexity if we apply MRME-TAD without post-processing for the lowest resolution subbands. In addition, MRME-TAD is still more efficient than FSBMA and the computational efficiency of MRME-TAD is quite higher for a scalable video coder compared to a FSBMA-based scalable video coder.

6.6 Conclusions

In this chapter, we propose a new algorithm for wavelet-based multi-resolution motion estimation (MRME) using temporal aliasing detection (TAD). This method improves MSE performance between motion compensated and current images compared to conventional MRME when temporal aliasing is severe. In addition, we can preserve the original signal without using an anti-aliasing filter. From experiments, we show that MRME-TAD algorithm can detect aliasing with some confidence and improve MSE compared to conventional MRME algorithm. When the motion is large/complex or well matched with our motion modeling, MRME-TAD algorithm improves MSE with small increase in the number of bytes. This is caused by the incorrect reference MVs for high frequency subbands in conventional MRME algorithm. The incorrect reference MVs of conventional MRME algorithm in high frequency subbands can be settled in the local minima of MSE. Even if the motion is small, MRME-TAD algorithm gives better performance with little extra bytes since the number of aliasing mode blocks is low. Another advantage of MRME-TAD

algorithm is that our algorithm can be combined to any other MRME algorithms without problems since it is additional processing based on conventional MRME algorithm.

(a)

(b)

(c)

Figure 6.8. Translational flower garden images for globally slow and fast motions. (a) 55th frame; (2) 57th frame; (c) 61the frame

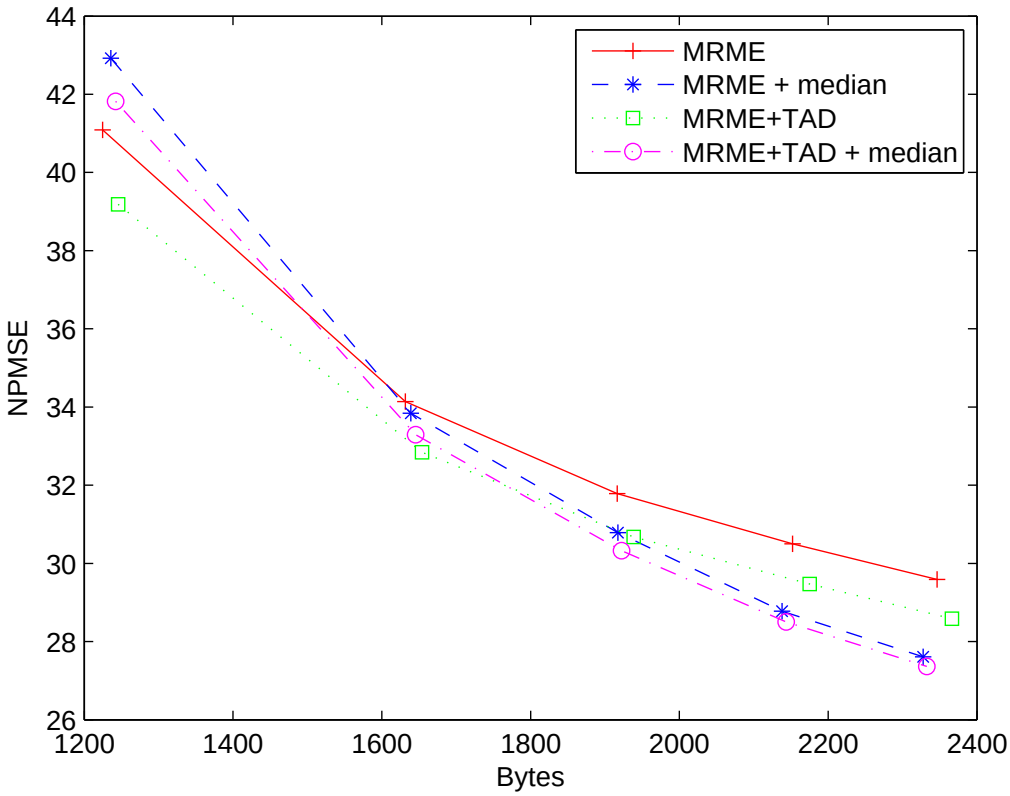


Figure 6.9. The performance comparison among MRME, MRME + median, MRME + TAD, and MRME + TAD + median schemes for football sequence. The search range from 3×3 to 11×11

(a)

(b)

(c)

Figure 6.10. Sample images of *Football* test sequence. (a) 16th frame; (2) 18th frame; (c) 22the frame

(a)

(b)

(c)

Figure 6.11. Sample images of *Mobile and calendar* test sequence. (a) 20th frame; (2) 25th frame; (c) 30the frame

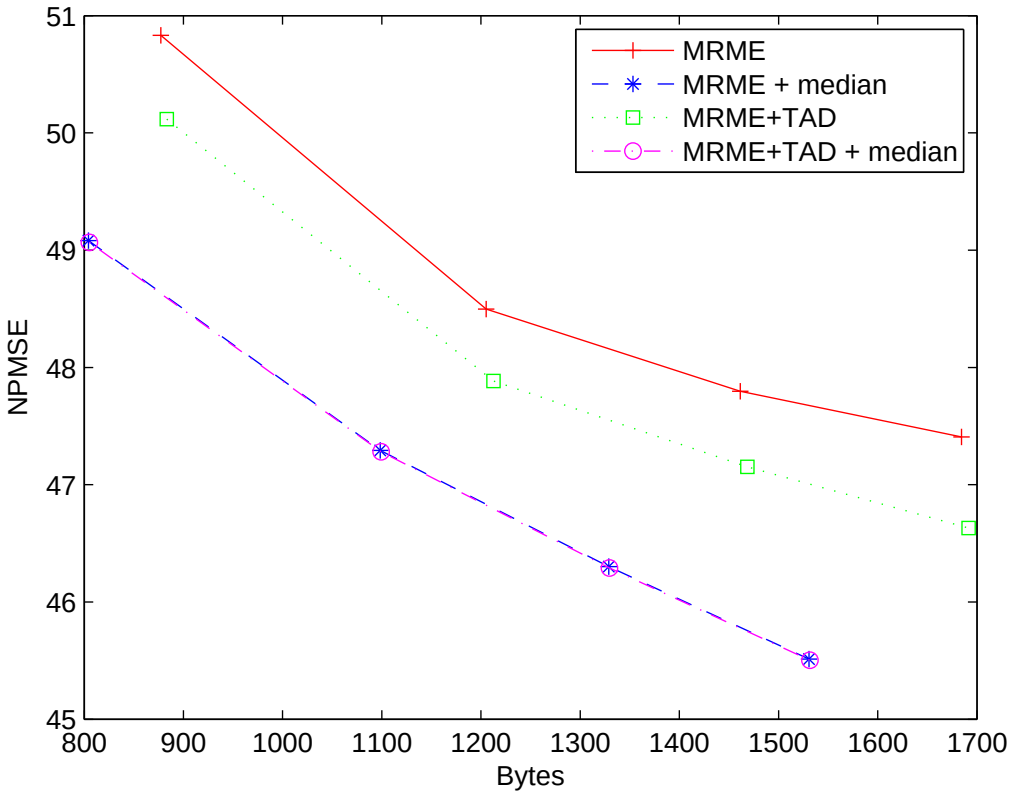


Figure 6.12. The performance comparison among MRME, MRME + median, MRME + TAD, and MRME + TAD + median schemes for mobile and calendar sequence. The search range from 3×3 **to** 9×9

CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH

Algorithm-based efficient approaches for motion estimation systems are proposed in terms of computational complexity, memory access, and/or performance. In addition, the algorithm performance is simulated using signal modeling with circuitry noise factors for CADSP system design. In this chapter, we will summarize our work by listing the contributions of this dissertation and addressing future research directions.

7.1 Summary of Contributions and Findings

The primary conclusions of the dissertation are summarized as follows:

1. In Chapter 3, signal modeling considering circuitry noise factors was presented for algorithm performance tests. For a better simulation, not only additive noise factors but also multiplicative noise factors are considered.

2. In Chapter 3, simulations of a gradient-based LS optical flow estimation algorithm under various sensor and system noises were performed to study the effects of linear-range, bit resolution, and noise. Based on our simulation, LS-OFE shows better performance for AGN than for other additive and multiplicative noise factors. This is reasonable since LS is the best estimator under additive Gaussian noise.

3. In Chapter 3, an architecture-level simulator was developed to achieve rapid algorithm verification for CADSP system design. It helps to predict the algorithm performance before the completion of analog and digital hardware implementation. In addition, noise tolerance can be reported to analog design team; this ensures that the analog system meets expected performance.

4. In Chapter 4, a checkerboard-type filtering algorithm was proposed for an efficient

gradient-based optical flow estimation (OFE) system. Checkerboard-type filtering (CBTF) is a filtering algorithm to implement fully parallel structure with data sharing (FPSDS) in a focal plane architecture and can move maximal amount of filtering to the imager to exploit the functionality of the STI. Checkerboard-type filtering with FPSDS is efficient in terms of computational complexity and the number of memory accesses compared to conventional filtering structure in STI-based system design. In addition, CBTF can implement both Simoncelli's filter and conventional prefitering and derivative filter steps efficiently on a focal plane imager.

5. In Chapter 4, the performance of FPSDS and conventional filtering structure (CS) for gradient-based OFE algorithms was compared. There are only slight performance differences for angular error performance between FPSDS and CS in a LS-based OFE algorithm. However, the optical flow density of FPSDS is between one-third and one-quarter of that of CS. This makes FPSDS profitable for applications without the requirement of high optical flow density.

6. In Chapter 5, a spatially recursive optical flow estimation (SR-OFE) framework using adaptive filtering was proposed. The sliding window RLS and adaptive filtering are employed to reduce the redundancies for calculating successive LS among adjacent pixels.

7. In Chapter 5, a LS-OFE framework using recursive matrix refinement (RMR-OFE) was suggested for efficient LS calculations. The efficiency is from the data reuse of linear system matrix among successive LS calculations.

8. In Chapter 5, LS-based efficient OFE algorithms using constant and affine motion models were presented based on the SR-OFE and RMR-OFE frameworks. The SR-OFE algorithm shows a better computational efficiency than the RMR-OFE algorithm when the number of update points is small enough for affine motion model. As

the number of update points increases, RMR-OFE algorithm outperforms SR-OFE algorithm in terms of computational complexity.

9. In Chapter 5, SR-OFE and RMR-OFE frameworks are shown to decrease computational complexity more as the number of motion modeling parameters increases because RMR-OFE efficiently reuse the data of linear system matrix and standard LS needs $O(n^3)$ and RLS requires $O(kn^2)$ computations, where n is the number of motion modeling parameters and k is the number of updated inputs per LS computation.

10. In Chapter 6, a new algorithm for wavelet-based multi-resolution motion estimation (MRME) using temporal aliasing detection (TAD) was proposed. This method improves motion compensation performance compared to conventional MRME when temporal aliasing is severe. In addition, we can preserve the original signal without using an anti-aliasing filter.

11. In Chapter 6, experiments showed that the MRME-TAD algorithm can detect aliasing with some confidence and improve MSE compared to conventional MRME. When the motion is large/complex or well matched with our motion modeling, MRME-TAD improves MSE with small increase in the number of bytes.

12. In Chapter 6, one important advantage of MRME-TAD algorithm is that it can be combined to any other conventional MRME algorithms without difficulty since it is merely a post-processing step for conventional MRME.

7.2 Directions for Future Research

Research always generates more questions than it answers. This section lists a few directions that others may with to take to follow-up on this work.

1. An architecture-level simulator was developed to achieve rapid algorithm verification for CADSP system design. To verify the efficiency in real system design, it would

be interesting to compare the design times of system implementation with or without our rapid algorithm verification method.

2. Rapid algorithm verification methodology was proposed for reducing hardware and software algorithm co-verification time of the system under CADSP design concept by using a CADSP imager simulator incorporating physical and architectural modelings. The work on algorithm partitioning between analog and digital domains presented in this dissertation is just a beginning. Future researchers may want to focus on methodologies and more general applications and systems.

3. The analysis of the effects of noise and bit resolution on the performance of an OFE as presented in this dissertation can be easily extended to other imaging systems and algorithms.

4. A checkerboard-type filtering (CBTF) algorithm was proposed for a efficient gradient-based optical flow estimation (OFE) system. The downside of CBTF is the low optical flow density than that of a LS-OFE algorithm based on conventional filtering. To increase optical flow density, efficient post-processing algorithm can be explored.

5. LS-based efficient OFE algorithms using constant and affine motion modelings were presented based on SR-OFE and RMR-OFE frameworks. We have been able to find parametric motion models with up to 12 parameters [6]. Since our frameworks can be applied to other parametric motion models, we can prove the computational efficiency for other parametric motion models.

6. SR-OFE and RMR-OFE frameworks were applied for non-iterative Lukas-Kanade algorithm. To improve the computational complexity, we can also apply our frameworks for original LK-OFE algorithm with proper modifications.

7. SR-OFE can be numerically sensitive because it is based on the normal equation. To improve the numerical sensitivity, a QR-based approach can be attempted.

8. A new algorithm for wavelet-based multi-resolution motion estimation (MRME) using temporal aliasing detection (TAD) was proposed. MRME-TAD has higher computational complexity than MRME; however, by exploiting correlation of the mode map among subbands, it should be possible to find a lower complexity approach.

9. Wavelet-based scalable video coding has advantages on spatial resolution and SNR scalabilities because of the wavelet spatial-temporal localization property. However, MRME-TAD algorithm is only a motion estimation algorithm. To fully analyze the impact of MRME-TAD, it should be incorporated into a full scalable video coder.

APPENDIX A

PROOF OF THE TEMPORAL ALIASING EQUATIONS BY THE OVERLAPPING OF TEMPORAL-SPATIAL SPECTRUM

$$i(x_o, t) = i(x_o - vt) = i\Big(- v\Big(t - \frac{x_o}{v}\Big)\Big). \tag{A.1}$$

To perform Fourier transform of $i(x_o, t)$,

$$\begin{aligned} I_{x_o}(f_t) &= \int i\Big(- v\Big(t - x_o/v\Big)\Big) e^{j2\pi t f_t} dt \\ &= \frac{1}{|v|} I\Big(- \frac{f_t}{v}\Big) e^{-j2\pi f_t x_o/v}. \end{aligned} \tag{A.2}$$

Eq A.2 is valid for non-aliasing case. If there are aliasing components, Eq A.2 should be modified as Eq 6.4.

APPENDIX B

PROOF OF THE ALIASING EQUATIONS BY SPATIAL DOWN-SAMPLING

If we use the same direction in LL band as reference MV,

$$X_1(w) = \frac{1}{2}\Big(H\Big(\frac{w}{2}\Big)X\Big(\frac{w}{2}\Big) + H\Big(\frac{w}{2}+\pi\Big)X\Big(\frac{w}{2}+\pi\Big)\Big)e^{-jw(v/2)}, \tag{B.1}$$

and

$$Y_1(w) = \frac{1}{2}\Big(H\Big(\frac{w}{2}\Big)X\Big(\frac{w}{2}\Big)e^{-j(w/2)v} + H\Big(\frac{w}{2}+\pi\Big)Y\Big(\frac{w}{2}+\pi\Big)e^{-j(w/2+\pi)v}\Big). \tag{B.2}$$

To calculate block matching operation, absolute difference between $X_1(w)$ and $Y_1(w)$ should be performed. It can be described as follows:

$$\begin{aligned} |X_1(w)-Y_1(w)| &= \Big|\frac{1}{2}H\Big(\frac{w}{2}+\pi\Big)X\Big(\frac{w}{2}+\pi\Big)\Big| \times \Big|(e^{-jw(v/2)} - e^{-j(w/2+\pi)v})\Big| \\ &= \Big|\frac{1}{2}H\Big(\frac{w}{2}+\pi\Big)X\Big(\frac{w}{2}+\pi\Big)e^{-jw(v/2)}\Big| \times \Big|(1-e^{-j\pi v})\Big| \end{aligned} \tag{B.3}$$

If we put $|\frac{1}{2}H(\frac{w}{2}+\pi)X(\frac{w}{2}+\pi)e^{-jw(v/2)}|$ as A,

$$|X_1(w) - Y_1(w)| = A|(1-e^{-j\pi v})|. \tag{B.4}$$

We can predict the worst and best cases from Eq. B.4. If v is odd number,

$$|X_1(w) - Y_1(w)| = 2A. \tag{B.5}$$

If v is even number,

$$|X_1(w) - Y_1(w)| = 0. \tag{B.6}$$

Therefore, we have perfect matching when v is even number. However, the worst performance can be obtained if v is odd number.

We can also derive the absolute difference equation for backward motion case. If we use the opposite direction in LL band as reference MV,

$$X_2(w) = \frac{1}{2}\Big(H\Big(\frac{w}{2}\Big)X\Big(\frac{w}{2}\Big) + H\Big(\frac{w}{2}+\pi\Big)X\Big(\frac{w}{2}+\pi\Big)\Big)e^{-jw(-v/2)}, \tag{B.7}$$

and

$$Y_1(w) = \frac{1}{2}\Big(H\Big(\frac{w}{2}\Big)X\Big(\frac{w}{2}\Big)e^{-j(w/2)v} + H\Big(\frac{w}{2}+\pi\Big)Y\Big(\frac{w}{2}+\pi\Big)e^{-j(w/2+\pi)v}\Big). \tag{B.8}$$

To calculate block matching operation, absolute difference between $X_2(w)$ and $Y_1(w)$ should be performed. It can be described as follows:

$$\begin{aligned}|X_2(w)-Y_1(w)| = \Big|&\frac{1}{2}H\Big(\frac{w}{2}\Big)X\Big(\frac{w}{2}\Big)(e^{jw(v/2)} - e^{-j(w/2)v}) \\ &+ \frac{1}{2}H\Big(\frac{w}{2}+\pi\Big)X\Big(\frac{w}{2}+\pi\Big) \times (e^{jw(v/2)} - e^{-j(w/2+\pi)v})\Big|\end{aligned} \tag{B.9}$$

By the triangular inequality, Eq. B will be as follows:

$$\begin{aligned}|X_2(w)-Y_1(w)| \le &\Big|\frac{1}{2}H\Big(\frac{w}{2}\Big)X\Big(\frac{w}{2}\Big)\Big| \times \Big|2j\sin\Big(\frac{w}{2}v\Big)\Big| \\ &+ \Big|\frac{1}{2}H\Big(\frac{w}{2}+\pi\Big)X\Big(\frac{w}{2}+\pi\Big)e^{jw(v/2)}\Big| \times \Big|(1-e^{-j(w+\pi)v})\Big|\end{aligned} \tag{B.10}$$

If we substitute $|\frac{1}{2}H(\frac{w}{2})X(\frac{w}{2})|$ with B and $|\frac{1}{2}H(\frac{w}{2}+\pi)X(\frac{w}{2}+\pi)e^{jw(v/2)}|$ with C,

$$|X_2(w) - Y_1(w)| \le B\Big|2j\sin\Big(\frac{w}{2}v\Big)\Big| + C\Big|(1-e^{-j(w+\pi)v})\Big| \tag{B.11}$$

If $B \ll C$ and v is odd number, we can have smaller absolute difference than Eq. B.4 by using temporal aliasing detection step.

Therefore, we can reduce temporal aliasing from spatial downsampling with fast temporal sampling compared to the motion vector by employing temporal aliasing detection algorithm.

REFERENCES

[1] Ackland, B. and Dickinson, A., "Camera on a chip," in *Proceedings of the 1996 IEEE International Solid-State Circuits Conference*, pp. 22–25, Feb. 1996.

[2] Anandan, P., "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, pp. 283–310, Jan 1989.

[3] Bandyopadhyay, A., *Matrix Transform Imager Architecture for On-Chip Low-Power Image Processing*. PhD thesis, Georgia Institute of Technology, 2004.

[4] Bandyopadhyay, A., Hasler, P., and Anderson, D. V., "A CMOS floating-point matrix transform imager," *IEEE Sensors*, vol. 5, no. 3, pp. 455–462, 2005.

[5] Barron, J. L., Fleet, D. J., and Beauchemin, S. S., "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, pp. 43–77, Feb. 1994.

[6] Barron, J. L. and Khurana, M., "Determining optical flow for large motions using parametric models in a hierarchical framework," in *Vision Interface (VI1997)*, (Kelowna, B.C.), May 1997.

[7] Battiti, R., Amaldi, E., and Koch, C., "Computing optical flow across multiple scales: An adpative coarse-to-fine strategy," *International Journal of Computer Vision*, vol. 6, no. 2, pp. 133–145, 1991.

[8] Bergen, J., Anandan, P., Hanna, K., and Hingorani, R., "Hierarchical model-based motion estimation," in *Proc. ECCV*, pp. 237–252, Springer-Verlag, 1992.

[9] Blanksby, A. and Loinaz, M. J., "Performance analysis of a color CMOS photogate image sensor," *IEEE Trans. on Electron Devices*, vol. 47, pp. 55–64, Jan 2000.

[10] Bloss, H. S., Ernst, J. D., Eirla, H., Schmoelz, S. C., Gick, S. K., and Lauxtermann, S., "High-speed-camera based on a CMOS active pixel sensor," in *Proc. SPIE*, vol. 3968, pp. 31–38, Feb 2000.

[11] Camus, T., "Real-time quantized optical flow," *J. Real-Time Imaging*, vol. 3, pp. 71–86, 1997.

[12] Christmas, W. J., "Filtering requirements for gradient-based optical flow measurement," *IEEE Trans. Image Processing*, vol. 9, pp. 1817–1820, Oct. 2000.

[13] Daud, T., Janesick, J., Evans, K., and Elliott, T., "Charge-coupled-device response to electron beam energies of less than 1 keV up 20 keV," *Optical Engineering*, vol. 26, pp. 686–691, Aug. 1987.

[14] Elad, M., Teo, P., and Hel-Or, Y., "Optimal filters for gradient-based motion estimation," in *IEEE International Conference on Computer Vision*, (Kerkyra, Corfu), Sept. 1999.

[15] Etienne-Cummings, R., der Spiegel, J. V., and Muller, P., "A focal plane visual motion measurement sensor," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 55–66, Jan. 1997.

[16] Farid, H. and Simoncelli, E. P., "Optimally rotation-equivariant directional derivative kernels," in *Proc 7th Int'l Conf Computer Analysis of Images and Patterns*, (Germany), Sep 1997.

[17] Farid, H. and Simoncelli, E. P., "Differentiation of discrete multi-dimensional signals," *IEEE Tran. on Image Processing*, vol. 13, no. 4, pp. 496–508, 2004.

[18] Fleet, D. J. and Langley, K., "Recursive filters for optical flow," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, pp. 61–67, Jan. 1995.

[19] Fossum, E. R., "Digital camera system on a chip," *IEEE Micro*, vol. 18, pp. 8–15, May-June 1998.

[20] Fossum, H. R., "CMOS image sensors: electronic camera on a chip," in *Proceedings of International Electron Devices Meeting*, (Washington, DC, USA), pp. 17–25, Dec. 1995.

[21] Frantz, G., "Digital signal processor trends," *IEEE Micro*, pp. 52–59, Nov. 2000.

[22] Hall, T. S., Twigg, C. M., Gray, J. D., Hasler, P., and Anderson, D. V., "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Tran. on Circuits and Systems*, vol. 52, no. 11, pp. 2298–2307, 2005.

[23] Hasler, P. and Anderson, D. V., "Cooperative analog-digital signal processing," in *IEEE International Conference on Acoustic, Speech, and Signal Processing*, 2002.

[24] Hasler, P., Bandyopadhyay, A., and Anderson, D. V., "High fill-factor imagers for neuromorphic processing enabled by floating-gate circuits," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 7, pp. 676–689, 2003.

[25] Hasler, P., Bandyopadhyay, A., and Smith, P., "A matrix transform imager allowing high-fill factor," in *IEEE International Symposium on Circuits and Systems*, vol. 3, 2002.

[26] Hatan, K. and *el al.*, "A 1/3inch 1.3mpixel sigle-layer electrode CCD with high-frame-rate skip mode," in *Proceedings of the 2000 International Solid State Circuits Conference*, (Washington, DC, USA), pp. 112–113, Dec. 2000.

[27] Hayes, M. H., *Statistical Digital Signal Processing and Modeling*. New York: John Wiley and Sons, 1996.

[28] Higgins, C. M., Deutschmann, R. A., and Koch, C., "Pulse-based 2-d motion sensors," *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing*, vol. 46, pp. 677–687, June 1999.

[29] Horn, B. K. P. and Schunck, B. G., "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[30] Itakura, K. and *et al.*, "A 1mm 50k.pixel it CCD image sensor for miniature camera system," in *Proceedings of the 1998 International Solid State Circuits Conference*, (San Francisco, CA, USA), pp. 180–181, Feb. 1998.

[31] Kay, S., *Fundamentals of Statistical Signal Processing*. Englewood Cliffs, New Jersey: Prentice–Hall, 1993.

[32] Kearney, J. K., Thompson, W. B., and Boley, D. L., "Optical flow estimation: An error analysis of gradient-based methods with local optimization," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 229–244, Mar. 1987.

[33] Kim, S., Rhee, S., Jeon, J. G., and Park, K. T., "Interframe coding using two-stage variable block-size multiresolution motion estimation and wavelet decomposition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 399–409, Aug. 1998.

[34] Lee, J., Lim, K., Song, B., and Ra, J., "A fast multi-resolution block matching algorithm and its LSI architecture for low bit-rate video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 1289–1301, Dec. 2001.

[35] Lee, T. and Anderson, D. V., "Behavioral simulation results of the optical flow estimation algorithm for CADSP sytem design," in *38th Asilomar Conf. Signals, Syst., and Comput.*, (Pacific Grove, CA), Nov 2004.

[36] Lee, T. and Anderson, D. V., "Checkerboard-type filtering for a low-power gradinet-based optical flow estimation system," in *IEEE International Conference on Image Processing*, (Atlanta, GA), Oct 2006.

[37] Lee, T. and Anderson, D. V., "Performance analysis of a correlation-based optical flow estimation algorithm under noisy environments," in *IEEE International Symposium on Circuits and Systems*, (Island of Kos), May 2006.

[38] Lee, T. and Anderson, D. V., "The wavelet-based multi-resolution motion estimation using temporal aliasing detection," in *Visual Communications and Image Processing*, (San Jose, CA), Jan 2007.

[39] Lee, T., Chiu, L. K., Anderson, D. V., Robucci, R., and Hasler, P., "Rapid algorithm verification for cooperative analog-digital imaging systems," in *IEEE International Midwest Symposium on Circuits and Systems*, (Montreal), Aug 2007.

[40] Li, W. and Salari, E., "Successive elimination algorithm for motion estimation," *IEEE Tran. on Image Processing*, vol. 4, no. 1, pp. 105–107, 1995.

[41] Lim, S., Apostolopoulos, J., and Gamal, A. E., "Optical flow estimation using temporally oversampled video," *IEEE Tran. on Image Processing*, vol. 14, no. 8, pp. 1074–1087, 2007.

[42] Lim, S. H., *Video processing applications of high speed* CMOS *image sensors*. PhD thesis, Stanford University, 2003.

[43] Liu, H., Chellappa, R., and Rosenfeld, A., "Accurate dense optical flow estimation using adaptive structure tensors and a parametric model," *IEEE Trans. Image Processing*, vol. 12, pp. 1170–1180, Oct. 2003.

[44] Liu, H., Hong, T. H., Herman, M., Camus, T., and Chellappa, R., "Accuracy vs efficiency trade-offs in optical flow algorithms," *Comp. Vision Image Understanding*, vol. 72, pp. 271–286, 1998.

[45] Liu, L.-K. and Feig, E., "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Tran. on Circuits Syst. Video Technol.*, vol. 6, pp. 419–423, Aug. 1996.

[46] Lucas, B. and Kanade, T., "An iterative image registration technique with an application to stereo vision," in *Proceedings of DARPA Image Understaing Workshop*, pp. 121–130, 1981.

[47] Mallat, S., "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.

[48] McCane, B., Novins, K., Crannitch, D., and Galvin, B., "On benchmarking optical flow," *Comp. Vision Image Understanding*, vol. 84, pp. 126–143, 2001.

[49] Mead, C. A., *Anlog VLSI and Neural Systems*. MA: Addison-Wesley, 1989.

[50] Netravali, A. N. and Haskell, B. G., *Digital Pictures Representation and Compression*. 1988.

[51] Oppenheim, A. V. and Schafer, R. W., *Discrete-Time Signal Processing*. Englewood Cliffs, New Jersey: Prentice–Hall, 1999.

[52] Park, H. and Kim, H., "Motion estimation using low-band-shift method for wavelet-based moving-picture coding," *IEEE Trans. Image Processing*, vol. 9, pp. 577–587, Apr. 2000.

[53] Po, L.-M. and Ma, W.-C., "A novel four-step seasrch algorithm for fast block motion estimation," *IEEE Tran. on Circuits Syst. Video Technol.*, vol. 6, pp. 313–317, Jun 1996.

[54] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes in C*. New York: Cambridge University Press, 1992.

[55] Rav-Acha, A. and Peleg, S., "Lukas-Kanade without iterative warping," in *IEEE International Conference on Image Processing*, (Atlanta), Oct 2006.

[56] Rhemann, C., "Region-based optical flow estimation with treatment of occlusions," Master's thesis, Vienna University of Technology, 2005.

[57] Robucci, R., "On chip error compensation, light adaptation, and image enhancement with a cmos transform image sensor," Master's thesis, Georgia Institute of Technology, 2005.

[58] Ryan, R. and *et al.*, "Compressive sensing on a cmos separable transform image sensor," in *IEEE International Conference on Acoustic, Speech, and Signal Processing*, 2008.

[59] Said, A. and Pearlman, W., "A new,fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.

[60] Sarpeshkar, R., *Efficient precise computation with noisy components: extrapolating from an electronic cochlea to the brain*. PhD thesis, California Institute of Technology, 1997.

[61] Sarpeshkar, R., Kramer, J., Indiveri, G., and Koch, C., "Analog VLSI architectures for motion processing: from fundamental limits to system applications," *Proc. IEEE*, vol. 84, pp. 969–987, July 1996.

[62] Shapiro, J., "Embedded image coding using zerotree of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

[63] Simoncelli, E. P., "Design of multi-dimensional derivative filters," in *IEEE International Conference on Image Processing*, 1994.

[64] Simoncelli, E., *Distributed Analysis and Representation of Visual Motion*. PhD thesis, MIT, 1993.

[65] Singh, A., *Optical Flow Computation: A Unified Perspective*. Los Alamitos, California, USA: IEEE Computer Society Press, 1991.

[66] Strang, G. and Nguyen, T., *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[67] Tekalp, A. M., *Digital Video Processing*. Englewood Cliffs, New Jersey: Prentice–Hall, 1995.

[68] Tian, H., *Noise analysis in cmos image sensors*. PhD thesis, Stanford University, 2000.

[69] Tsunashima, K., Stampleman, J. B., and Bove, V. M., "A scalable motion-compensated subband imgae coder," *IEEE Trans. Commun.*, vol. 42, pp. 1894–1901, Apr. 1994.

[70] Uz, K., Vetterli, M., and LeGall, D., "Interpolative multiresolution coding of advanced television and compatible subchannels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 86–99, Mar. 1991.

[71] Verri, A. and Poggio, T., "Against quantitative optical flow," in *IEEE International Conference on Computer Vision*, (London), June 1987.

[72] Vetterli, M. and Kovacevic, J., *Wavelets and Subband Coding*. Englewood Cliffs, New Jersey: Prentice–Hall, 1995.

[73] Wang, H.-S. and Mersereau, R. M., "Fast algorithms for the estimation of motion vectors," *IEEE Tran. on Image Processing*, vol. 8, no. 3, pp. 435–438, 1999.

[74] Waxman, A. M. and Wohn, K., "Contour evolution, neighborhood deformation, and global image flow: planar surfaces in motion," *Int. J. of Robotics Res.*, vol. 4, no. 3, pp. 95–108, 1985.

[75] Weber, J. and Malik, J., "Robust computation of optical flow in multi-scale differential framework," *International Journal of Computer Vision*, vol. 14, no. 1, pp. 67–81, 1995.

[76] Witten, I. H., Neal, R. M., and Cleary, J. G., "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, June. 1987.

[77] Wong, H.-S., "Technology and device scaling considerations for CMOS imagers," *IEEE Transactions on Electron Devices*, vol. 43, pp. 2131–2141, Dec. 1996.

[78] Yang, X. and Ramchandran, K., "Scalable wavelet video coding using aliasing-reduced hierarchical motion compensation," *IEEE Trans. Image Processing*, vol. 9, pp. 778–791, May 2000.

[79] Yoo, H., *Low-power audio input enhancements for portable devices*. PhD thesis, Georgia Institute of Technology, 2005.

[80] Zan, J., Ahmad, M., and Swamy, M., "New techniques for multi-resolution motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 793–802, Sept. 2002.

[81] Zhang, Y.-Q. and Zafar, S., "Motion-compensated wavelet transform coding for color video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 285–296, Sept. 1992.

[82] Zhu, S. and Ma, K.-K., "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Tran. on Image Processing*, vol. 9, pp. 287–290, Feb. 2000.